(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0135611
(43) 공개일자 2024년09월11일

(51) 국제특허분류(Int. Cl.)
H04N 19/70 (2014.01) *H04N 19/107* (2014.01)
H04N 19/124 (2014.01) *H04N 19/184* (2014.01)
H04N 19/463 (2014.01)
(52) CPC특허분류
H04N 19/70 (2015.01)
H04N 19/107 (2015.01)
(21) 출원번호 10-2024-7023703
(22) 출원일자(국제) 2022년11월08일
심사청구일자 없음
(85) 번역문제출일자 2024년07월15일
(86) 국제출원번호 PCT/US2022/079490
(87) 국제공개번호 WO 2023/132992
국제공개일자 2023년07월13일
(30) 우선권주장
63/266,615 2022년01월10일 미국(US)
(뒷면에 계속)

(71) 출원인
광동 오포 모바일 텔레커뮤니케이션즈 코포레이션 리미티드
중국, 광동 523860, 동관, 창안, 우샤, 하이빈 로드, 넘버 18
(72) 발명자
간, 조나단
미국, 캘리포니아 94303, 팰로앨토, 이스트 베이 쇼어 로드 2479, 스위트 110 이노피크 테크놀로지 인코퍼레이티드 씨/오
유, 유에
미국, 캘리포니아 94303, 팰로앨토, 이스트 베이 쇼어 로드 2479, 스위트 110 이노피크 테크놀로지 인코퍼레이티드 씨/오
유, 하오핑
미국, 캘리포니아 94303, 팰로앨토, 이스트 베이 쇼어 로드 2479, 스위트 110 이노피크 테크놀로지 인코퍼레이티드 씨/오
(74) 대리인
특허법인이룸리온

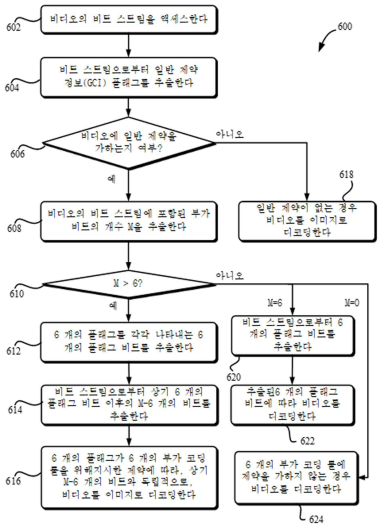
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **비디오 코딩의 일반 제약 정보를 전송**

(57) 요 약

일부 실시예에서, 비디오 디코더는 비디오의 비트 스트림으로부터 비디오를 디코딩한다. 비디오 디코더는 비디오의 비트 스트림으로부터 부가 비트 카운트 M을 추출하고, 여기서, 부가 비트 카운트 M은 비디오의 비트 스트림에 포함된 부가 일반 제약 정보(GCI) 비트의 개수를 지시한다. 부가 비트는 플래그 비트를 포함하고, 플래그 비트는 각 부가 코딩 툴이 비디오의 경우 제약되는 것을 각각 지시하고, 부가 비트 카운트의 기대값은 0 또는 6이다. 추출된 부가 비트 카운트 M이 6보다 큰 것으로 결정한 경우, 디코더는 비트 스트림에서의 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출한다. 더 나아가, 디코더는 추출된 M-6 개의 비트에 의존하지 않고, 6 개의 플래그 비트가 각 부가 코딩 툴을 위해 각각 규정한 제약에 적어도 부분적으로 기반하여, 비트 스트림의 나머지 부분을 이미지로 디코딩한다.

대 표 도 - 도6



(52) CPC특허분류
H04N 19/124 (2015.01)
H04N 19/184 (2015.01)
H04N 19/463 (2015.01)

(30) 우선권주장
63/266,616 2022년01월10일 미국(US)
63/266,765 2022년01월13일 미국(US)

명 세 서

청구범위

청구항 1

비디오 디코딩 방법으로서,

비디오의 비트 스트림으로부터 부가 비트 카운트 M을 추출하는 단계 - 상기 부가 비트 카운트 M은 상기 비디오의 비트 스트림에 포함된 부가 일반 제약 정보(GCI) 비트의 개수를 지시하고, 상기 부가 비트는 플래그 비트를 포함하며, 상기 플래그 비트는 상기 비디오의 경우 제약되는 각 부가 코딩 툴을 지시하고, 상기 부가 비트 카운트의 기대값은 0 또는 6임 - ;

추출된 상기 부가 비트 카운트 M이 6보다 큰 것으로 결정된 것에 응답하여, 상기 비트 스트림에서의 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출하는 단계; 및

추출된 상기 M-6 개의 비트에 의존하지 않고, 상기 6 개의 플래그 비트가 각 부가 코딩 툴을 위해 각각 규정한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 단계를 포함하는 것을 특징으로 하는 비디오 디코딩 방법.

청구항 2

제1항에 있어서,

상기 비디오 디코딩 방법은,

상기 M-6 개의 비트를 추출하기 전, 상기 비디오의 비트 스트림으로부터 6 개의 플래그를 각각 대표하는 상기 6 개의 플래그 비트를 추출하는 단계 - 상기 6 개의 플래그는 상기 비디오의 경우 제약되는 6 개의 부가 코딩 툴을 각각 지시함 - ; 및

상기 6 개의 플래그가 상기 6 개의 부가 코딩 툴을 위해 지시한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 단계를 더 포함하는 것을 특징으로 하는 비디오 디코딩 방법.

청구항 3

제2항에 있어서,

상기 6 개의 플래그는,

상기 비디오의 픽처가 인트라 랜덤 액세스 포인트(IRAP) 픽처 또는 점진적 디코딩 리프레시(GDR) 픽처로 제한되는 것을 지시하는 제1 플래그;

확장 변환 정밀도를 제약하는지 여부를 지시하는 제2 플래그;

명시적 라이스 파라미터 전송을 제약하는지 여부를 지시하는 제3 플래그;

상기 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도를 지시하는 제4 플래그;

이전의 변환 유닛에 기반한 이진화를 위한 라이스 파라미터 유도를 초기화하는지 여부를 지시하는 제5 플래그; 및

변환 유닛 중 마지막 0이 아닌 레벨의 위치를 디코딩할 경우, 범위 내 출력층 세트(OlsInScope)에서의 픽처에 제약을 가하는지 여부를 지시하는 제6 플래그를 포함하는 것을 특징으로 하는 비디오 디코딩 방법.

청구항 4

제3항에 있어서,

상기 6 개의 플래그가 상기 6 개의 부가 코딩 툴을 위해 지시한 제약에 적어도 부분적으로 기반하여, 상기 비디

오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 단계는,

상기 제1 플래그가 1인 것에 기반하여, 하나 또는 복수 개의 출력층 세트에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 결정하고, 상기 하나 또는 복수 개의 출력층 세트에서의 상기 GDR 픽처 또는 상기 IRAP 픽처를 디코딩하는 단계;

상기 제2 플래그가 1인 것에 기반하여, 확장 변환 정밀도가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 sps_extended_precision_flag를 0과 같게 설정하여 확장 동적 범위를 사용하지 않는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 단계;

상기 제3 플래그가 1인 것에 기반하여, 명시적 라이스 파라미터 전송이 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 단계;

상기 제4 플래그가 1인 것에 기반하여, 상기 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 단계;

상기 제5 플래그가 1인 것으로 결정한 것에 기반하여, 이전의 변환 유닛 상태에 기반한 이진화를 위한 라이스 파라미터 유도에 대한 초기화가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 경우 이전의 변환 유닛 상태에 기반하여 라이스 파라미터를 초기화하지 않는 경우, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 단계; 또는

상기 제6 플래그가 1인 것에 기반하여, 마지막 유효 계수의 좌표는 슬라이스의 각 변환 블록의 왼쪽 위 코너에 대해 코딩된 것으로 결정하고, 상기 마지막 유효 계수의 디코딩된 좌표를 상기 슬라이스의 각 변환 블록에 대한 왼쪽 위 코너인 것으로 해석하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 단계 중 하나 또는 복수 개를 포함하는 것을 특징으로 하는 비디오 디코딩 방법.

청구항 5

제3항에 있어서,

상기 비디오 디코딩 방법은,

상기 비트 스트림에 상기 제1 플래그가 존재하지 않는 것으로 결정하고, 상기 제1 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 단계;

상기 비트 스트림에 상기 제2 플래그가 존재하지 않는 것으로 결정하고, 상기 제2 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 단계;

상기 비트 스트림에 상기 제3 플래그가 존재하지 않는 것으로 결정하고, 상기 제3 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 단계;

상기 비트 스트림에 상기 제4 플래그가 존재하지 않는 것으로 결정하고, 상기 제4 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 단계;

상기 비트 스트림에 상기 제5 플래그가 존재하지 않는 것으로 결정하고, 상기 제5 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 단계; 또는

상기 비트 스트림에 상기 제6 플래그가 존재하지 않는 것으로 결정하고, 상기 제6 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 단계 중 하나 또는 복수 개를 더 포함하는 것을 특징으로 하는 비디오 디코딩 방법.

청구항 6

제1항에 있어서,

상기 부가 비트 카운트 M을 추출하기 전, 상기 비디오 디코딩 방법은,

상기 비디오의 비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출하는 단계; 및

상기 GCI 플래그의 값에 기반하여, 상기 비디오에 하나 또는 복수 개의 일반 제약을 가한 것으로 결정하는 단계

를 더 포함하는 것을 특징으로 하는 비디오 디코딩 방법.

청구항 7

제6항에 있어서,

상기 GCI 플래는 상기 비디오의 네트워크 데이터 패킷, 상기 비디오의 비디오 파라미터 세트 또는 상기 비디오의 시퀀스 파라미터 세트로부터 추출된 것임을 특징으로 하는 비디오 디코딩 방법.

청구항 8

비 일시적 컴퓨터 판독 가능한 매체로서,

상기 비 일시적 컴퓨터 판독 가능한 매체에는 프로그램 코드가 저장되어 있고, 상기 프로그램 코드는 하나 또는 복수 개의 처리 기기에 의해 실행되어 동작을 수행하며, 상기 동작은,

비디오의 비트 스트림으로부터 부가 비트 카운트 M을 추출하는 것 - 상기 부가 비트 카운트 M은 상기 비디오의 비트 스트림에 포함된 부가 일반 제약 정보(GCI) 비트의 개수를 지시하고, 상기 부가 비트는 플래그 비트를 포함하며, 상기 플래그 비트는 상기 비디오의 경우 제약되는 각 부가 코딩 툴을 지시하고, 상기 부가 비트 카운트의 기대값은 0 또는 6임 - ;

추출된 상기 부가 비트 카운트 M이 6보다 큰 것으로 결정된 것에 응답하여, 상기 비트 스트림에서의 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출하는 것; 및

추출된 상기 M-6 개의 비트에 의존하지 않고, 상기 6 개의 플래그 비트가 각 부가 코딩 툴을 위해 각각 규정한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것을 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 9

제8항에 있어서,

상기 동작은,

상기 M-6 개의 비트를 추출하기 전, 상기 비디오의 비트 스트림으로부터 6 개의 플래그를 각각 대표하는 상기 6 개의 플래그 비트를 추출하는 것 - 상기 6 개의 플래그는 상기 비디오의 경우 제약되는 6 개의 부가 코딩 툴을 각각 지시함 - ; 및

상기 6 개의 플래그가 상기 6 개의 부가 코딩 툴을 위해 지시한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것을 더 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 10

제9항에 있어서,

상기 6 개의 플래그는,

상기 비디오의 픽처가 인트라 랜덤 액세스 포인트(IRAP) 픽처 또는 점진적 디코딩 리프레시(GDR) 픽처로 제한되는 것을 지시하는 제1 플래그;

확장 변환 정밀도를 제약하는지 여부를 지시하는 제2 플래그;

명시적 라이스 파라미터 전송을 제약하는지 여부를 지시하는 제3 플래그;

상기 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도를 지시하는 제4 플래그;

이전의 변환 유닛에 기반한 이진화를 위한 라이스 파라미터 유도를 초기화하는지 여부를 지시하는 제5 플래그; 및

변환 유닛 중 마지막 0이 아닌 레벨의 위치를 디코딩할 경우, 범위 내 출력층 세트(OlsInScope)에서의 픽처에 제약을 가하는지 여부를 지시하는 제6 플래그를 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 11

제10항에 있어서,

상기 6 개의 플래그가 상기 6 개의 부가 코딩 툴을 위해 지시한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것은,

상기 제1 플래그가 1인 것에 기반하여, 하나 또는 복수 개의 출력층 세트에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 결정하고, 상기 하나 또는 복수 개의 출력층 세트에서의 상기 GDR 픽처 또는 상기 IRAP 픽처를 디코딩하는 것;

상기 제2 플래그가 1인 것에 기반하여, 확장 변환 정밀도가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 sps_extended_precision_flag를 0과 같게 설정하여 확장 동적 범위를 사용하지 않는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것;

상기 제3 플래그가 1인 것에 기반하여, 명시적 라이스 파라미터 전송이 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것;

상기 제4 플래그가 1인 것에 기반하여, 상기 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것;

상기 제5 플래그가 1인 것으로 결정한 것에 기반하여, 이전의 변환 유닛 상태에 기반한 이진화를 위한 라이스 파라미터 유도에 대한 초기화가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 경우 이전의 변환 유닛 상태에 기반하여 라이스 파라미터를 초기화하지 않는 경우, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것; 또는

상기 제6 플래그가 1인 것에 기반하여, 마지막 유효 계수의 좌표는 슬라이스의 각 변환 블록의 왼쪽 위 코너에 대해 코딩된 것으로 결정하고, 상기 마지막 유효 계수의 디코딩된 좌표를 상기 슬라이스의 각 변환 블록에 대한 왼쪽 위 코너인 것으로 해석하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것 중 하나 또는 복수 개를 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 12

제10항에 있어서,

상기 동작은,

상기 비트 스트림에 상기 제1 플래그가 존재하지 않는 것으로 결정하고, 상기 제1 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제2 플래그가 존재하지 않는 것으로 결정하고, 상기 제2 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제3 플래그가 존재하지 않는 것으로 결정하고, 상기 제3 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제4 플래그가 존재하지 않는 것으로 결정하고, 상기 제4 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제5 플래그가 존재하지 않는 것으로 결정하고, 상기 제5 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것; 또는

상기 비트 스트림에 상기 제6 플래그가 존재하지 않는 것으로 결정하고, 상기 제6 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것 중 하나 또는 복수 개를 더 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 13

제8항에 있어서,

상기 부가 비트 카운트 M을 추출하기 전, 상기 동작은,

상기 비디오의 비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출하는 것; 및

상기 GCI 플래그의 값에 기반하여, 상기 비디오에 하나 또는 복수 개의 일반 제약을 가한 것으로 결정하는 것을 더 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 14

제13항에 있어서,

상기 GCI 플래는 상기 비디오의 네트워크 데이터 패킷, 상기 비디오의 비디오 파라미터 세트 또는 상기 비디오의 시퀀스 파라미터 세트로부터 추출된 것임을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 15

시스템으로서,

처리 기기; 및

상기 처리 기기에 통신적으로 커플링된 비 일시적 컴퓨터 판독 가능한 매체를 포함하고; 상기 처리 기기는 상기 비 일시적 컴퓨터 판독 가능한 매체에 저장된 프로그램 코드를 실행함으로써, 동작을 실행하기 위한 것이고, 상기 동작은,

비디오의 비트 스트림으로부터 부가 비트 카운트 M을 추출하는 것 - 상기 부가 비트 카운트 M은 상기 비디오의 비트 스트림에 포함된 부가 일반 제약 정보(GCI) 비트의 개수를 지시하고, 상기 부가 비트는 플래그 비트를 포함하며, 상기 플래그 비트는 상기 비디오의 경우 제약되는 각 부가 코딩 툴을 지시하고, 상기 부가 비트 카운트의 기대값은 0 또는 6임 - ;

추출된 상기 부가 비트 카운트 M이 6보다 큰 것으로 결정된 것에 응답하여, 상기 비트 스트림에서의 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출하는 것; 및

추출된 상기 M-6 개의 비트에 의존하지 않고, 상기 6 개의 플래그 비트가 각 부가 코딩 툴을 위해 각각 규정한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것을 포함하는 것을 특징으로 하는 시스템.

청구항 16

제15항에 있어서,

상기 동작은,

상기 M-6 개의 비트를 추출하기 전, 상기 비디오의 비트 스트림으로부터 6 개의 플래그를 각각 대표하는 상기 6 개의 플래그 비트를 추출하는 것 - 상기 6 개의 플래그는 상기 비디오의 경우 제약되는 6 개의 부가 코딩 툴을 각각 지시함 - ; 및

상기 6 개의 플래그가 상기 6 개의 부가 코딩 툴을 위해 지시한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것을 더 포함하는 것을 특징으로 하는 시스템.

청구항 17

제16항에 있어서,

상기 6 개의 플래그는,

상기 비디오의 픽처가 인트라 랜덤 액세스 포인트(IRAP) 픽처 또는 점진적 디코딩 리프레시(GDR) 픽처로 제한되는 것을 지시하는 제1 플래그;

확장 변환 정밀도를 제약하는지 여부를 지시하는 제2 플래그;

명시적 라이스 파라미터 전송을 제약하는지 여부를 지시하는 제3 플래그;

상기 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도를 지시하는 제4 플래그;

이전의 변환 유닛에 기반한 이진화를 위한 라이스 파라미터 유도를 초기화하는지 여부를 지시하는 제5 플래그; 및

변환 유닛 중 마지막 0이 아닌 레벨의 위치를 디코딩할 경우, 범위 내 출력층 세트(OlsInScope)에서의 픽처에 제약을 가하는지 여부를 지시하는 제6 플래그를 포함하는 것을 특징으로 하는 시스템.

청구항 18

제17항에 있어서,

상기 6 개의 플래그가 상기 6 개의 부가 코딩 툴을 위해 지시한 제약에 적어도 부분적으로 기반하여, 상기 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것은,

상기 제1 플래그가 1인 것에 기반하여, 하나 또는 복수 개의 출력층 세트에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 결정하고, 상기 하나 또는 복수 개의 출력층 세트에서의 상기 GDR 픽처 또는 상기 IRAP 픽처를 디코딩하는 것;

상기 제2 플래그가 1인 것에 기반하여, 확장 변환 정밀도가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 sps_extended_precision_flag를 0과 같게 설정하여 확장 동적 범위를 사용하지 않는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것;

상기 제3 플래그가 1인 것에 기반하여, 명시적 라이스 파라미터 전송이 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것;

상기 제4 플래그가 1인 것에 기반하여, 상기 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것;

상기 제5 플래그가 1인 것으로 결정한 것에 기반하여, 이전의 변환 유닛 상태에 기반한 이진화를 위한 라이스 파라미터 유도에 대한 초기화가 제약되는 것으로 결정하고, 상기 범위 내 출력층 세트(OlsInScope)에서의 픽처의 경우 이전의 변환 유닛 상태에 기반하여 라이스 파라미터를 초기화하지 않는 경우, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것; 또는

상기 제6 플래그가 1인 것에 기반하여, 마지막 유효 계수의 좌표는 슬라이스의 각 변환 블록의 왼쪽 위 코너에 대해 코딩된 것으로 결정하고, 상기 마지막 유효 계수의 디코딩된 좌표를 상기 슬라이스의 각 변환 블록에 대한 왼쪽 위 코너인 것으로 해석하는 것을 통해, 상기 비디오의 비트 스트림의 나머지 부분을 디코딩하는 것 중 하나 또는 복수 개를 포함하는 것을 특징으로 하는 시스템.

청구항 19

제17항에 있어서,

상기 동작은,

상기 비트 스트림에 상기 제1 플래그가 존재하지 않는 것으로 결정하고, 상기 제1 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제2 플래그가 존재하지 않는 것으로 결정하고, 상기 제2 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제3 플래그가 존재하지 않는 것으로 결정하고, 상기 제3 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제4 플래그가 존재하지 않는 것으로 결정하고, 상기 제4 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것;

상기 비트 스트림에 상기 제5 플래그가 존재하지 않는 것으로 결정하고, 상기 제5 플래그의 값이 0인 것으로 추론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것; 또는

상기 비트 스트림에 상기 제6 플래그가 존재하지 않는 것으로 결정하고, 상기 제6 플래그의 값이 0인 것으로 추

론하여, 상응한 코딩 툴에 제약을 가하지 않은 것으로 지시하는 것 중 하나 또는 복수 개를 더 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

청구항 20

제15항에 있어서,

상기 부가 비트 카운트 M을 추출하기 전, 상기 동작은,

상기 비디오의 네트워크 데이터 패킷, 상기 비디오의 비디오 파라미터 세트 또는 상기 비디오의 시퀀스 파라미터 세트로부터 일반 제약 정보(GCI) 플래그를 추출하는 것을 통해, 상기 비디오의 비트 스트림으로부터 상기 GCI 플래그를 추출하는 것; 및

상기 GCI 플래그의 값에 기반하여, 비디오에 하나 또는 복수 개의 일반 제약을 가한 것으로 결정하는 것을 더 포함하는 것을 특징으로 하는 비 일시적 컴퓨터 판독 가능한 매체.

발명의 설명

기 술 분 야

[0001] 관련 출원에 대한 상호 참조

[0002] 본 출원은 2022년 1월 10일에 제출한 출원 명칭이 “Signaling Methods for General Constraints Information for Video Coding(비디오 코딩을 위한 일반 제약 정보의 전송 방법)” 인 제63/266,615호 미국 임시 출원, 2022년 1월 10일에 제출한 출원 명칭이 “Initialization Method for General Constraint Information Flags for Video Coding(비디오 코딩을 위한 일반 제약 정보 플래그의 초기화 방법)” 인 제63/266,616호 미국 임시 출원, 2022년 1월 13일에 제출한 출원 명칭이 “Signaling and Initialization Methods for General Constraints Information for Video Coding(비디오 코딩을 위한 일반 제약 정보의 전송 및 초기화 방법)” 인 제 63/266,765호 미국 출원의 우선권을 주장한다. 위의 전부 내용은 인용되어 본 출원에 결합된다.

[0003] 본 발명은 일반적으로 비디오 처리에 관한 것이다. 구체적으로, 본 발명은 비디오 코딩의 일반 제약 정보의 전송(signaling) 및 초기화에 관한 것이다.

배 경 기 술

[0004] 어디나 존재하는, 스마트폰, 태블릿 컴퓨터 및 컴퓨터 등과 같은 촬영 기기는, 비디오 또는 이미지의 촬영으로 하여금 매우 편이하도록 한다. 그러나, 짧은 비디오에 사용되는 데이터 양도 매우 클 수 있다. 비디오 코딩 기술(비디오 코딩 및 디코딩을 포함함)은 비디오 데이터를 더욱 작은 사이즈로 압축하는 것을 허용함으로써, 다양한 비디오를 저장 및 전송하는 것을 허용한다. 비디오 코딩은, 디지털 텔레비전 브로드캐스트, 인터넷, 모바일 네트워크에서의 비디오 전송, 실시간 애플리케이션(예컨대 비디오 채팅, 비디오 회의), DVD, 블루-레이 디스크 등과 같은 다양한 영역에 이미 광범위하게 적용되고 있다. 비디오를 저장하기 위한 저장 공간 및 비디오를 전송하기 위한 네트워크 대역폭 소모 중 적어도 하나를 감소하기 위해, 비디오 코딩 방안의 효율을 향상시킬 것을 요망한다.

발명의 내용

과제의 해결 수단

[0005] 일부 실시예는 비디오 코딩을 위한 일반 제약 정보의 전송 및 초기화에 관한 것이다. 하나의 예에서, 비디오 디코딩 방법은, 비디오의 비트 스트림으로부터 부가 비트 카운트 M을 추출하는 단계 - 부가 비트 카운트 M은 비디오의 비트 스트림에 포함된 부가 일반 제약 정보(GCI) 비트의 개수를 지시하고, 부가 비트는 플래그 비트를 포함하며, 플래그 비트는 각 부가 코딩 툴이 상기 비디오의 경우 제약되는 것을 각각 지시하며, 부가 비트 카운트의 기대값은 0 또는 6임 - ; 추출된 부가 비트 카운트 M이 6보다 큰 것으로 결정된 경우, 비트 스트림에서의 6개의 플래그 비트 이후의 M-6 개의 비트를 추출하는 단계; 및 추출된 M-6 개의 비트에 의존하지 않고, 상기 6개의 플래그 비트가 각 부가 코딩 툴을 위해 각각 규정한 제약에 적어도 부분적으로 기반하여, 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 단계를 포함한다.

[0006] 다른 하나의 예에서, 비 일시적 컴퓨터 판독 가능한 매체로서, 상기 비 일시적 컴퓨터 판독 가능한 매체에는 프

로그램 코드가 저장되어 있고, 프로그램 코드는 하나 또는 복수 개의 처리 기기에 의해 실행되어 동작을 수행할 수 있다. 동작은, 비디오의 비트 스트림으로부터 부가 비트 카운트 M을 추출하는 것 - 부가 비트 카운트 M은 비디오의 비트 스트림에 포함된 부가 일반 제약 정보(GCI) 비트의 개수를 지시하고, 부가 비트는 플래그 비트를 포함하며, 플래그 비트는 각 부가 코딩 툴이 상기 비디오의 경우 제약되는 것을 각각 지시하며, 부가 비트 카운트의 기대값은 0 또는 6임 - ; 추출된 부가 비트 카운트 M이 6보다 큰 것으로 결정된 경우, 비트 스트림에서의 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출하는 것; 및 추출된 M-6 개의 비트에 의존하지 않고, 상기 6 개의 플래그 비트가 각 부가 코딩 툴을 위해 각각 규정한 제약에 적어도 부분적으로 기반하여, 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것을 포함한다.

[0007] 또 다른 하나의 예에서, 시스템은 처리 기기; 및 상기 처리 기기와 통신적으로 커플링된 비 일시적 컴퓨터 판독 가능한 매체를 포함한다. 상기 처리 기기는 비 일시적 컴퓨터 판독 가능한 매체에 저장된 프로그램 코드를 실행함으로써, 동작을 실행하기 위한 것이다. 동작은, 비디오의 비트 스트림으로부터 부가 비트 카운트 M을 추출하는 것 - 부가 비트 카운트 M은 비디오의 비트 스트림에 포함된 부가 일반 제약 정보(GCI) 비트의 개수를 지시하고, 부가 비트는 플래그 비트를 포함하며, 플래그 비트는 각 부가 코딩 툴이 상기 비디오의 경우 제약되는 것을 각각 지시하며, 부가 비트 카운트의 기대값은 0 또는 6임 - ; 추출된 부가 비트 카운트 M이 6보다 큰 것으로 결정된 경우, 비트 스트림에서의 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출하는 것; 및 추출된 M-6 개의 비트에 의존하지 않고, 상기 6 개의 플래그 비트가 각 부가 코딩 툴을 위해 각각 규정한 제약에 적어도 부분적으로 기반하여, 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는 것을 포함한다.

[0008] 이러한 설명성 실시예를 언급하는 것은 본 발명을 제한하거나 정의하기 위한 것이 아니라, 예를 제공하여 본 발명을 이해하는데 도움을 주기 위해서이다. 구체적인 실시 형태에서 또한 다른 실시예를 논의하고 추가로 설명하였다.

도면의 간단한 설명

[0009] 도면을 참조하여 아래의 구체적인 실시 형태를 열독할 경우, 본 발명의 특징, 실시예 및 장점을 더욱 잘 이해할 수 있다.

도 1은 본 문에서 제출한 실시예를 구현하기 위한 비디오 인코더의 예의 블록도를 도시한다.

도 2는 본 문에서 제출한 실시예를 구현하기 위한 비디오 디코더의 예의 블록도를 도시한다.

도 3은 본 발명의 일부 실시예에 따른 비디오에서의 픽처에 대해 코딩 트리 유닛 분할을 수행하는 예를 설명한다.

도 4는 본 발명의 일부 실시예를 따른 코딩 트리 유닛에 대해 코딩 유닛 분할을 수행하는 예를 설명한다.

도 5는 본 발명의 일부 실시예를 따른 비디오 디코딩 과정의 예를 설명한다.

도 6은 본 발명의 일부 실시예를 따른 비디오 디코딩 과정의 다른 하나의 예를 설명한다.

도 7은 본 발명의 일부 실시예를 따른 비디오 디코딩 과정의 다른 하나의 예를 설명한다.

도 8은 본 발명의 일부 실시예에 사용될 수 있는 컴퓨팅 시스템의 예를 설명한다.

발명을 실시하기 위한 구체적인 내용

[0010] 각 실시예는 비디오 코딩을 위한 일반 제약 정보의 전송 및 초기화를 제공한다. 전술한 바와 같이, 점점 더욱 많은 비디오 데이터가 생성, 저장 및 전송된다. 유익한 것은, 비디오 코딩 기술의 효율도 향상시키고, 비디오 코딩의 안정성도 향상시켜, 디코더측에서 비디오 신호를 성공적으로 디코딩할 수 있도록 한다. 비디오 디코딩의 안정성에 관련된 문제는 불호환 및 불일치 문제를 포함한다. 비디오 코딩 기술이 발전됨에 따라, 비교적 새로운 비디오 코딩 표준도 따라서 생성된다. 그 중의 하나의 비디오 코딩 표준은 다기능 비디오 코딩 표준 제1 버전이고, 국제 표준화 조직(ISO)의 “ISO/IEC 23090-3:2021 정보 기술－－몰입형 미디어의 코딩 표현법－－제 3 부분: 다기능 비디오 코딩”의 명의 및 국제 전기 통신 연맹(ITU)의 “ITU-T H.26 제안(08/2020): 다기능 비디오 코딩”의 명의로 연합하여 발표하였다. 본 발명에서, 다기능 비디오 코딩 표준 제1 버전은 “VVC 제1 버전” 또는 “VVCv1”로 지칭될 수 있다. VVC 제1 버전은 다기능 비디오 코딩 표준의 제2 버전에 의해 대채되었고, 제2 버전은 ISO의 “ISO/IEC 23090-3:2022 정보 기술－－몰입형 미디어의 코딩 표현법－－제3 부분:다기능 비디오 코딩”의 명의 및 ITU의 “ITU-T H.266 제안(04/2022): 다기능 비디오 코딩”의 명의로 연합하여 발표한다. 본

발명에서, 다기능 비디오 코딩 표준 제2 버전은 “VVC 제2 버전” 또는 “VVCv2”로 지칭될 수 있다. 이번 버전 비디오 코딩 표준을 사용하여 코딩된 비디오 신호가 새로운 버전 비디오 코딩 표준을 사용하는 비디오 디코더에 의해 성공적으로 디코딩될 수 있도록 하기 위해, 비디오 코딩 방안은 이전 버전의 코딩 표준과 하위 호환될 수 있도록 설계되어야 한다. 그러나, 상이한 버전의 비디오 코딩 표준 사이의 엄중한 불호환 문제로 인해, VVC 제2 버전의 현재 초안에서 사용되는 일반 제약 정보의 전송은 비디오 디코딩의 비 동기화를 초래한다. 또한, VVC 제2 버전의 현재 초안에서, 일반 제약 정보와 관련된 일반 제약 플래그는 특정된 경우 정의되지 않을 수 있음으로써, 디코더 구현 방식이 불명확하고 불일치할 수 있다. 본 문에서 설명된 다양한 실시예는 비디오 코딩을 위한 일반 제약 정보의 전송 및 초기화 방법을 인입하는 것을 통해 이러한 문제를 해결함으로써, 비디오 코딩의 안정성을 향상시킨다.

[0011] VVC 표준에서, 일반 제약 정보(GCI) 신택스 구조 general_constraints_info()는 비트 스트림의 특정 제약 속성을 지시하기 위한 것이다. GCI는 제약 플래그 및 비 플래그 신택스 요소의 리스트를 포함한다. 이진 플래그 gci_present_flag는 GCI 신택스 요소가 존재하는지 여부를 규정하기 위한 것이다. 특정된 실시예에서, VVC 제2 버전 비트 스트림이 일반 제약 정보를 전송하고(즉, gci_present_flag의 값이 1임), VVC 제2 버전 일반 제약 정보가 가능하게 제약되는 N 개의 부가 코딩 툴로 구성되면, 상기 N 개의 부가 코딩 툴에 대응되는 신택스 요소 gci_num_additional_bits의 값은 0 또는 N으로만 설정될 수 있다. gci_num_additional_bits가 0으로 설정되면, 상기 N 개의 부가 코딩 툴의 일반 제약 플래그를 전송하지 않는다. gci_num_additional_bits가 N으로 설정되면, 비트 스트림에서의 다음 N 개의 비트는 상기 N 개의 부가 코딩 툴의 일반 제약 플래그를 전송하는데 사용된다. 하나의 예에서, N은 6으로 설정된다.

[0012] 일부 예에서, VVC 제2 버전 비트 스트림은 gci_num_additional_bits을 0 또는 N 이외의 값으로 설정하는 것을 허용하지 않는다. 그러나, VVC 제2 버전 디코더는 gci_num_additional_bits을 0 또는 N 이외의 값으로 설정한 일반 제약 정보를 여전히 처리할 수 있다. 예컨대, gci_num_additional_bits을 M으로 설정할 수 있고, M은 0보다 크고 N보다 작으며, 또는 M은 N보다 크다. M이 0보다 크고 N보다 작으면, gci_num_additional_bits 신택스 요소를 디코딩한 후, 디코더는 비트 스트림으로부터 M 개의 비트를 추출하고 버린다. M이 N보다 크면, gci_num_additional_bits 신택스 요소를 디코딩한 후, 디코더는 비트 스트림으로부터 N 개의 비트를 추출하고, 상기 N 개의 비트를 N 개의 부가 코딩 툴의 일반 제약 플래그로 해석한다. 다음, 디코더는 비트 스트림으로부터 M-N 개의 비트를 다시 추출하고 버린다. 다른 예에서, VVC 제2 버전 디코더는 gci_num_additional_bits을 0보다 크지만 N보다 작은 값으로 설정한 일반 제약 정보를 처리할 필요가 없다. 합법적인 VVC 제2 버전 비트 스트림은 gci_num_reserved_bits의 값을 0 또는 N으로만 설정할 수 있다. VVC 미래 버전의 비트 스트림은 gci_num_additional_bits을 0부터 N 사이의 값으로 설정하는 것을 허용하지 않는다.

[0013] 일부 실시예에서, 일반 제약 정보 플래그를 초기화하여 상기 디코더 구현 방식이 불명확하고 불일치한 문제를 해결한다. 이러한 실시예에서, gci_present_flag가 1과 같고 gci_num_additional_bits가 0과 같을 경우, general_constraints_info()는 일반 제약 정보 플래그와 관련된 코딩 툴에 제약을 가하지 않는다. 플래그 값이 0인 것이 제약이 없는 것으로 지시하는 예에서, 플래그가 존재하지 않을 경우, 일반 제약 정보 플래그의 값은 0과 같은 것으로 추론된다.

[0014] 본 발명에서 설명된 실시예는 VVC 제2 버전에서 부가 코딩 툴의 일반 제약 플래그의 전송 및 추론 방법을 제공한다. 기존 기술과 상이한 것은, 본 발명에서 설명된 방법에 의해 생성된 고레벨 신택스 비트 스트림은 VVC 제1 버전 디코더와 호환되고, VVC 제1 버전 디코더와 VVC 제2 버전 디코더 행위 사이의 비 동기화를 초래하지 않고 디코딩될 수 있다. 본 발명에서 설명된 추론 규칙은 VVC 제2 버전 GCI 신택스 요소의 VVC 제2 버전 디코딩 행위의 불명확성을 소거하였다. 이러한 기술은 각 비디오 코딩 표준에서의 유효 코딩 툴이 될 수 있다.

[0015] 도면을 참조하면, 도 1은 비디오 인코더(100)의 예의 블록도를 도시하고, 상기 비디오 인코더(100)는 본 문에서 제공한 실시예를 구현하기 위한 것이다. 도 1에 도시된 예에서, 비디오 인코더(100)는 분할 모듈(112), 변환 모듈(114), 양자화 모듈(115), 역양자화 모듈(118), 역변환 모듈(119), 인루프 필터 모듈(120), 인트라 예측 모듈(126), 인터 예측 모듈(124), 모션 추정 모듈(122), 디코딩 픽처 버퍼(130) 및 엔트로피 코딩 모듈(116)을 포함한다.

[0016] 비디오 인코더(100)의 입력은 입력 비디오(102)이고, 여기서 일련의 픽처(프레임 또는 이미지로 지칭될 수도 있음)를 포함한다. 블록에 기반한 비디오 인코더에서, 각 픽처에 대해, 비디오 인코더(100)는 분할 모듈(112)을 사용하여 픽처를 블록(104)으로 분할하고, 각 블록은 복수 개의 픽셀을 포함한다. 이러한 블록은 매크로 블록, 코딩 트리 유닛, 코딩 유닛, 예측 유닛 및 예측 블록 중 적어도 하나일 수 있다. 하나의 픽처는 상이한 크기의

블록을 포함할 수 있고, 비디오 중 상이한 픽처의 블록 분할도 상이할 수 있다. 각 블록은 인트라 예측, 인터 예측 및 인트라 및 인터의 혼합 예측과 같은 상이한 예측을 사용하여 코딩될 수 있다.

[0017] 일반적으로, 비디오 신호의 첫 번째 픽처는 인트라 코딩 픽처이고, 인트라 예측만 사용하여 코딩된다. 인트라 예측 모드에서, 동일한 픽처 중 이미 코딩된 데이터만 사용하여 픽처의 블록을 예측한다. 인트라 코딩된 픽처는 다른 픽처의 정보를 사용하지 않고 디코딩될 수 있다. 인트라 예측을 실행하기 위해, 도 1에 도시된 비디오 인코더(100)는 인트라 예측 모듈(126)을 사용할 수 있다. 인트라 예측 모듈(126)은 동일한 픽처의 인접 블록의 재구성 블록(136)에서의 재구성 샘플을 사용하여 인트라 예측 블록(예측 블록(134))을 생성하기 위한 것이다. 상기 블록을 위해 선택된 인트라 예측 모드에 따라 인트라 예측을 실행한다. 다음, 비디오 인코더(100)는 블록(104)과 인트라 예측 블록(134) 사이의 차이값을 계산한다. 상기 차이값은 잔차 블록(106)으로 지칭된다.

[0018] 블록에서의 리던던시를 추가로 소거하기 위해, 변환 모듈(114)은 블록에서의 샘플을 변환하는 것을 통해, 잔차 블록(106)을 변환 도메인에 변환한다. 변환의 예는 이산 코사인 변환(DCT) 또는 이산 사인 변환(DST)을 포함하지만 이에 한정되지 않는다. 변환된 후의 값은 변환 계수로 지칭될 수 있고, 변환 도메인에서의 잔차 블록을 나타낸다. 일부 예에서, 변환 모듈(114)의 변환을 거치지 않고 직접 잔차 블록을 양자화할 수 있다. 이러한 모드를 변환 스킵 모드로 지칭한다.

[0019] 비디오 인코더(100)는 양자화 모듈(115)을 추가로 사용하여 변환 계수를 양자화하여, 양자화 계수를 획득할 수 이다. 양자화는 샘플을 양자화 스텝 사이즈로 나눈 다음, 반올림하는 것을 포함하지만, 역양자화는 양자화값에 양자화 스텝 사이즈를 곱한 것이다. 이러한 양자화 과정을 스칼라 양자화로 지칭한다. 양자화는 (변환된 후 또는 변환되지 않은) 비디오 샘플의 동적 범위를 감소함으로써, 더욱 적은 비트로 비디오 샘플을 나타내기 위한 것이다.

[0020] 블록 내의 계수/샘플의 양자화는 독립적으로 완료될 수 있으며, 기존의 일부 비디오 압축 표준(예컨대 H.264 및 HEVC)에서 이러한 양자화 방법을 사용한다. 하나의 N*M 블록에 대해, 특정된 스캔 순서를 사용하여 블록의 2 차원 계수를 1 차원 어레이로 전환하여, 계수 양자화 및 코딩을 수행할 수 있다. 블록 내의 계수의 양자화는 스캔 순서 정보를 이용할 수 있다. 예컨대, 블록 중 주어진 계수의 양자화는 스캔 순서 중 이전 양자화값의 상태에 따라 결정될 수 있다. 코딩 효율을 추가로 향상시키기 위해, 복수 개의 양자화기를 사용할 수 있다. 어느 양자화기를 사용하여 현재 계수를 양자화할지는, 코딩/디코딩 스캔 순서 중 현재 계수 이전의 정보에 따라 결정된다. 이러한 양자화 방법을 종속 양자화로 지칭한다.

[0021] 양자화 정도는 양자화 스텝 사이즈를 통해 조정할 수 있다. 예컨대, 스칼라 양자화에 대해, 상이한 양자화 스텝 사이즈를 사용하여 비교적 정밀하거나 비교적 조악한 양자화를 구현할 수 있다. 비교적 작은 양자화 스텝 사이즈는 비교적 정밀한 양자화에 대응되고, 비교적 큰 양자화 스텝 사이즈는 비교적 조악한 양자화에 대응된다. 양자화 스텝 사이즈는 양자화 파라미터(QP)를 통해 지시할 수 있다. 양자화 파라미터는 비디오의 코딩 비트 스트림에서 제공되어, 비디오 디코더로 하여금 양자화 파라미터에 액세스하고 적용하여 디코딩을 수행할 수 있도록 한다.

[0022] 다음, 엔트로피 코딩 모듈(116)은 양자화 샘플을 코딩하여, 비디오 신호의 크기를 추가로 감소시킨다. 엔트로피 코딩 모듈(116)은 양자화 샘플에 엔트로피 코딩 알고리즘을 적용하기 위한 것이다. 일부 예에서, 양자화 샘플은 이진 빈으로 이진화되고, 코딩 알고리즘은 상기 이진 빈을 비트로 추가로 압축한다. 이진화 방법의 예는 트렁케이티드 라이스(TR) 및 제한된 k차 지수 골롬(EGk) 조합의 이진화와, k차 지수 골롬 이진화를 포함하지만 이에 한정되지 않는다. 엔트로피 코딩 알고리즘의 예는 가변 길이 코딩(VLC) 방안, 컨텍스트 적응 VLC 방안(CAVLC), 산술 코딩 방안, 이진화, 컨텍스트 적응 이진 산술 코딩(CABAC), 신택스에 기반한 컨텍스트 적응 이진 산술 코딩(SBAC), 확률 구간 분할 엔트로피(PIPE) 코딩 또는 다른 엔트로피 코딩 기술을 포함하지만 이에 한정되지 않는다. 엔트로피 코딩된 데이터는 출력 코딩 비디오(132)의 비트 스트림에 추가된다.

[0023] 전술한 바와 같이, 인접 블록으로부터의 재구성 블록(136)은 픽처의 블록의 인트라 예측에 사용된다. 하나의 블록의 재구성 블록(136)을 생성하는 것은 상기 블록의 재구성 잔차를 계산하는 것에 관련된다. 재구성 잔차는 상기 블록의 양자화 잔차에 역양자화 및 역변환을 적용하는 것을 통해 결정될 수 있다. 역양자화 모듈(118)은 양자화 샘플에 역양자화를 적용하여, 탈양자화 계수를 획득하기 위한 것이다. 역양자화 모듈(118)은 양자화 모듈(115)과 동일한 양자화 스텝 사이즈를 사용하는 것을 통해, 양자화 모듈(115)에 적용되는 양자화 방안과 반대의 방안을 적용한다. 역변환 모듈(119)은 변환 모듈(114)에 적용되는 변환의 역변환을 역DCT 또는 역DST와 같은 탈양자화 샘플에 적용하기 위한 것이다. 역변환 모듈(119)의 출력은 픽셀 도메인 중 블록의 재구성 잔차이다. 재구성 잔차를 블록의 예측 블록(134)에 추가하여, 픽셀 도메인에서의 재구성 블록(136)을 획득할 수 있다. 변환

스킵된 블록에 대해, 역변환 모듈(119)을 적용하지 않는다. 탈양자화 샘플은 블록의 재구성 잔차이다.

[0024] 첫 번째 인트라 예측 픽처 이후의 후속 픽처에서의 블록은 인터 예측 또는 인트라 예측을 사용하여 코딩될 수 있다. 인터 예측에서, 픽처에서의 블록의 예측은 하나 또는 복수 개의 이전 이미 코딩된 비디오 픽처로부터 얻은 것이다. 인터 예측을 실행하기 위해, 비디오 인코더(100)는 인터 예측 모듈(124)을 사용한다. 인터 예측 모듈(124)은 모션 추정 모듈(122)에 의해 제공된 모션 추정에 기반하여 블록에 대해 모션 보상을 수행하기 위한 것이다.

[0025] 모션 추정 모듈(122)은 현재 픽처의 현재 블록(104)과 디코딩 참조 픽처(108)를 비교하여, 모션 추정을 수행한다. 디코딩 참조 픽처(108)는 디코딩 픽처 버퍼(130)에 저장된다. 모션 추정 모듈(122)은 디코딩 참조 픽처(108)로부터 현재 블록과 가장 매칭되는 참조 블록을 선택한다. 모션 추정 모듈(122)은 참조 블록의 위치(예컨대 x, y 좌표)와 현재 블록의 위치 사이의 오프셋을 추가로 식별한다. 상기 오프셋은 모션 벡터(MV)로 지칭되고, 선택된 참조 블록과 함께 인터 예측 모듈(124)에 제공된다. 특정된 경우, 복수 개의 디코딩 참조 픽처(108)에서의 현재 블록을 위해 복수 개의 참조 블록을 식별한다. 따라서, 복수 개의 모션 벡터를 생성하고, 상응한 참조 블록과 함께 인터 예측 모듈(124)에 제공된다.

[0026] 인터 예측 모듈(124)은 모션 벡터 및 다른 인터 예측 파라미터를 사용하여 모션 보상을 실행하여, 현재 블록의 예측, 즉, 인터 예측 블록(134)을 생성한다. 예컨대, 모션 벡터에 기반하여, 인터 예측 모듈(124)은 상응한 참조 픽처에서 모션 벡터가 가리키는 예측 블록을 로케이팅할 수 있다. 하나 이상의 예측 블록이 존재하면, 이러한 예측 블록을 일정한 가중치로 합병하여, 현재 블록의 예측 블록(134)을 생성한다.

[0027] 인터 예측 블록의 경우, 비디오 인코더(100)는 블록(104)에서 인터 예측 블록(134)을 감하여, 잔차 블록(106)을 생성할 수 있다. 상기 인트라 예측 블록의 잔차와 동일한 방식을 통해, 잔차 블록(106)에 대해 변환, 양자화 및 엔트로피 코딩을 수행할 수 있다. 마찬가지로, 잔차에 대해 역양자화, 역변환을 수행한 다음, 상응한 예측 블록(134)과 결합하여, 인터 예측 블록의 재구성 블록(136)을 획득할 수 있다.

[0028] 모션 추정을 위한 디코딩 픽처(108)를 획득하기 위해, 인루프 필터 모듈(120)로 재구성 블록(136)을 처리할 수 있다. 인루프 필터 모듈(120)은 스무딩하게 픽셀 전이함으로써, 비디오 품질을 향상시키기 위한 것이다. 인루프 필터 모듈(120)은, 디블록킹 필터, 샘플링 적응 오프셋(SAO) 필터, 적응 인루프 필터(ALF) 등과 같은 하나 또는 복수 개의 인루프 필터를 구현하기 위한 것일 수 있다.

[0029] 도 2는 비디오 디코더(200)의 예를 설명하고, 상기 비디오 디코더(200)는 본 문에서 제공한 실시예를 구현하기 위한 것이다. 비디오 디코더(200)는 비트 스트림에서의 코딩 비디오(202)를 처리하고 디코딩 픽처(208)를 생성한다. 도 2에 도시된 예에서, 비디오 디코더(200)는 엔트로피 디코딩 모듈(216), 역양자화 모듈(218), 역변환 모듈(219), 인루프 필터 모듈(220), 인트라 예측 모듈(226), 인터 예측 모듈(224) 및 디코딩 픽처 버퍼(230)를 포함한다.

[0030] 엔트로피 디코딩 모듈(216)은 코딩 비디오(202)에 대해 엔트로피 디코딩을 수행하기 위한 것이다. 엔트로피 디코딩 모듈(216)은 양자화 계수, 코딩 파라미터(인트라 예측 파라미터 및 인터 예측 파라미터를 포함함) 및 다른 정보를 디코딩한다. 일부 예에서, 엔트로피 디코딩 모듈(216)은 코딩 비디오(202)의 비트 스트림을 이진 표현으로 디코딩한 다음, 이진 표현을 계수의 양자화 레벨로 전환한다. 엔트로피 디코딩된 계수 레벨은 역양자화 모듈(218)에 의해 역양자화된 다음, 역변환 모듈(219)에 의해 픽셀 도메인에 역변환된다. 역양자화 모듈(218) 및 역변환 모듈(219)의 기능은 각각 도 1에 대해 설명된 역양자화 모듈(118) 및 역변환 모듈(119)과 유사하다. 역변환된 잔차 블록은 상응한 예측 블록(234)에 추가되어, 재구성 블록(236)을 생성할 수 있다. 변환 스킵된 블록에 대해, 역변환 모듈(219)을 적용하지 않는다. 역양자화 모듈(118)에 의해 생성된 탈양자화 샘플은 재구성 블록(236)을 생성하는데 사용된다.

[0031] 특정 블록의 예측 블록(234)은 상기 블록의 예측 모드에 기반하여 생성된다. 상기 블록의 코딩 파라미터가 상기 블록이 인트라 예측된 것임으로 지시하면, 동일한 픽처 중 참조 블록의 재구성 블록(236)을 인트라 예측 모듈(226)에 입력하여, 상기 블록의 예측 블록(234)을 생성할 수 있다. 상기 블록의 코딩 파라미터가 상기 블록이 인터 예측된 것임으로 지시하면, 인터 예측 모듈(224)에 의해 예측 블록(234)이 생성된다. 인트라 예측 모듈(226) 및 인터 예측 모듈(224)의 기능은 각각 도 1에서의 인트라 예측 모듈(126) 및 인터 예측 모듈(124)과 유사하다.

[0032] 도 1에 대해 전술한 바와 같이, 인터 예측은 하나 또는 복수 개의 참조 픽처에 관련된다. 비디오 디코더(200)는 인루프 필터 모듈(220)을 참조 픽처의 재구성 블록에 적용하는 것을 통해, 참조 픽처의 디코딩 픽처(208)를 생

성한다. 디코딩 픽처(208)는 디코딩 픽처 버퍼(230)에 저장되어, 인터 예측 모듈(224) 및 출력에 사용되도록 한다.

[0033] 도 3을 참조하면, 도 3은 본 발명의 일부 실시예에 따라, 비디오에서의 픽처에 대해 코딩 트리 유닛 분할을 수행하는 예를 설명한다. 도 1 및 도 2에 대해 전술한 바와 같이, 비디오의 픽처를 코딩하기 위해, 픽처는 도 3에 도시된 바와 같은 VVC에서의 CTU(코딩 트리 유닛)(302)와 같은 블록으로 분할된다. 예컨대, CTU(302)는 128x128 픽셀의 블록일 수 있다. 순서(예컨대 도 3에 도시된 순서)에 따라 CTU를 처리한다. 일부 예에서, 도 4에 도시된 바와 같이, 픽처에서의 각 CTU(302)는 하나 또는 복수 개의 CU(코딩 유닛)(402)로 분할 될 수 있고, CU(402)는 예측 유닛 또는 변환 유닛(TU)으로 추가로 분할되어, 예측 및 변환에 사용될 수 있다. 상이한 코딩 방안에 따라, CTU(302)는 상이한 방식으로 CU(402)로 분할될 수 있다. 예컨대, VVC에서, CU(402)는 직사각형 또는 정사각형일 수 있고, 또한 예측 유닛 또는 변환 유닛으로 추가로 분할되지 않고 코딩될 수 있다. 각 CU(402)는 루트 CTU(302) 만큼 클 수 있고, 4x4 블록 만큼 작은 루트 CTU(302)의 세분화일 수도 있다. 도 4에 도시된 바와 같이, VVC 중 CTU(302)로부터 CU(402)까지의 분할은 쿼드트리 분할, 이진 트리 분할 또는 삼진 트리 분할일 수 있다. 도 4에서, 실선은 쿼드트리 분할을 지시하고, 점선은 이진 트리 또는 삼진 트리 분할을 지시한다.

[0034] *VVC 제1 버전(VVCv1)에서의 일반 제약 정보*

[0035] VVC 제1 버전에서, 일반 제약 정보(GCI) 신택스 구조 general_constraints_info()는 비트 스트림의 특정 제약 속성을 지시하기 위한 것이다. GCI는 제약 플래그 및 비 플래그 신택스 요소의 리스트를 포함한다. 이진 플래그 gci_present_flag는 GCI 신택스 요소가 존재하는지 여부를 규정하기 위한 것이다. gci_present_flag가 1과 같으면 general_constraints_info() 신택스 구조에 GCI 신택스 요소가 존재하는 것으로 규정한다. gci_present_flag가 0과 같으면 general_constraints_info() 신택스 구조에 GCI 필드가 존재하는 것으로 규정하고, general_constraint_info() 신택스 구조는 임의의 제약을 가하지 않는다.

[0036] 다양한 콘텐츠에서 고레벨 신택스로 일반 제약 정보를 전송할 수 있다. 예컨대, nal_unit_type을 13(즉, DCI_NUT을 nal_unit_type의 명칭으로 사용함)으로 설정한 네트워크 추상화 레이어(NAL) 데이터 패킷과 같은, 디코딩 능력 정보만 포함하는 네트워크 데이터 패킷에서 GCI를 전송할 수 있고, 상기 데이터 패킷은 디코딩 능력 정보만 캐리한다. 또는, 비디오 파라미터 세트 또는 시퀀스 파라미터 세트에서 GCI를 전송할 수 있다.

[0037] GCI 신택스 구조의 목적은 비트 스트림을 디코딩하는데 필요한 기능과 관련된 구성 정보를 발견할 수 있고, 상호 운용 포인트를 전송하는 것을 허용하는 것이며, 이러한 상호 운용 포인트가 가한 제한은 프로파일, 계층 및 레벨(PTL)의 규정을 초과하고, 그 입도는 이전의 비디오 코딩 표준에 의해 허용되는 입도보다 더욱 미세하다. 서브 프로파일과 유사하게, GCI 신택스 구조의 사용은 VVC 프로파일의 모든 기능을 지원하지 않지만 특정 애플리케이션 요구를 만족하는 디코더 구현 방식을 위해 상호 운용성을 정의할 수 있도록 한다. 디코더 구현 방식은 GCI 신택스 요소를 심사하여, 비트 스트림이 특정 기능을 사용하는 것을 피하는지 여부를 체크할 수 있음으로써, 디코딩 과정을 구성하는 방법을 결정하고, 비트 스트림이 디코더에 의해 디코딩될 수 있는지 여부를 식별한다. VVC 프로파일을 지원하는 모든 기능의 디코더 구현 방식은 GCI 신택스 요소값을 무시할 수 있으며, 그 원인은 이러한 디코더는 지시된 PTL에 부합되는 임의의 비트 스트림을 디코딩할 수 있기 때문이다.

[0038] VVC 제1 버전에서 규정된 일반 제약 정보 신택스 구조는 아래와 같이 정의된다.

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
gci_all_layers_independent_constraint_flag	u(1)
gci_one_au_only_constraint_flag	u(1)
/* 픽처 포맷(picture format) */	
gci_sixteen_minus_max_bitdepth_constraint_idc	u(4)
gci_three_minus_max_chroma_format_constraint_idc	u(2)
/* NAL 유닛 타입 관련(NAL unit type related) */	
gci_no_mixed_nalu_types_in_pic_constraint_flag	u(1)
gci_no_trail_constraint_flag	u(1)
gci_no_stsa_constraint_flag	u(1)
gci_no_rasl_constraint_flag	u(1)
gci_no_radl_constraint_flag	u(1)
gci_no_idr_constraint_flag	u(1)

[0039]

gci_no_cra_constraint_flag	u(1)
gci_no_gdr_constraint_flag	u(1)
gci_no_aps_constraint_flag	u(1)
gci_no_idr_rpl_constraint_flag	u(1)
/* 타일, 슬라이스, 서브 픽처 분할(tile, slice, subpicture partitioning) */	
gci_one_tile_per_pic_constraint_flag	u(1)
gci_pic_header_in_slice_header_constraint_flag	u(1)
gci_one_slice_per_pic_constraint_flag	u(1)
gci_no_rectangular_slice_constraint_flag	u(1)
gci_one_slice_per_subpic_constraint_flag	u(1)
gci_no_subpic_info_constraint_flag	u(1)
/* CTU 및 블록 분할(CTU and block partitioning) */	
gci_three_minus_max_log2_ctu_size_constraint_idc	u(2)
gci_no_partition_constraints_override_constraint_flag	u(1)
gci_no_mtt_constraint_flag	u(1)
gci_no_qtbtt_dual_tree_intra_constraint_flag	u(1)
/* 인트라(intra) */	
gci_no_palette_constraint_flag	u(1)
gci_no_ibc_constraint_flag	u(1)
gci_no_isp_constraint_flag	u(1)

[0040]

gci_no_mrl_constraint_flag	u(1)
gci_no_mip_constraint_flag	u(1)
gci_no_cclm_constraint_flag	u(1)
/* 인터(inter) */	
gci_no_ref_pic_resampling_constraint_flag	u(1)
gci_no_res_change_in_clvs_constraint_flag	u(1)
gci_no_weighted_prediction_constraint_flag	u(1)
gci_no_ref_wraparound_constraint_flag	u(1)
gci_no_temporal_mvp_constraint_flag	u(1)
gci_no_sbtmvp_constraint_flag	u(1)
gci_no_amvr_constraint_flag	u(1)
gci_no_bdof_constraint_flag	u(1)
gci_no_smvd_constraint_flag	u(1)
gci_no_dmvr_constraint_flag	u(1)
gci_no_mmvd_constraint_flag	u(1)
gci_no_affine_motion_constraint_flag	u(1)
gci_no_prof_constraint_flag	u(1)
gci_no_bcw_constraint_flag	u(1)
gci_no_ciip_constraint_flag	u(1)
gci_no_gpm_constraint_flag	u(1)
/* 변환, 양자화, 잔차(transform, quantization, residual) */	

[0041]

gci_no_luma_transform_size_64_constraint_flag	u(1)
gci_no_transform_skip_constraint_flag	u(1)
gci_no_bdpcm_constraint_flag	u(1)
gci_no_mts_constraint_flag	u(1)
gci_no_lfnst_constraint_flag	u(1)
gci_no_joint_cbcr_constraint_flag	u(1)
gci_no_sbt_constraint_flag	u(1)
gci_no_act_constraint_flag	u(1)
gci_no_explicit_scaling_list_constraint_flag	u(1)
gci_no_dep_quant_constraint_flag	u(1)
gci_no_sign_data_hiding_constraint_flag	u(1)
gci_no_cu_qp_delta_constraint_flag	u(1)
gci_no_chroma_qp_offset_constraint_flag	u(1)
/* 인루프 필터(loop filter) */	
gci_no_sao_constraint_flag	u(1)
gci_no_alf_constraint_flag	u(1)
gci_no_ccalf_constraint_flag	u(1)
gci_no_lmcs_constraint_flag	u(1)
gci_no_ladf_constraint_flag	u(1)
gci_no_virtual_boundaries_constraint_flag	u(1)
gci_num_reserved_bits	u(8)
for(i = 0; i < gci_num_reserved_bits; i++)	

[0042]

gci_reserved_zero_bit[i]	u(1)
}	
while(!byte_aligned( ))	
gci_alignment_zero_bit	f(1)
}	

[0043]

[0044] 일반 제약 플래그의 존재는 gci_present_flag의 값에 따라 결정된다. gci_present_flag의 값이 1일 경우, 비트

스트림에 일반 제약 플래그가 존재한다. gci_present_flag의 값이 0일 경우, 비트 스트림에 일반 제약 플래그가 존재하지 않는다.

[0045] VVCv1에 의해 정의된 일반 제약 플래그 이외, 신택스 요소 gci_num_reserved_bits는 또한 부가 일반 제약 플래그를 규정하였다. gci_num_reserved_bits는 8 비트의 무부호 정수이고, 그 값은 일반 제약 신택스 구조 중 전송된 부가 비트수를 지시한다. VVC 규정에서, 비트 스트림으로부터 이러한 부가 비트(신택스 요소 gci_reserved_zero_bit[i]로 지칭됨)를 추출하고 버린다. 이러한 규정은 VVCv1 디코더로 하여금 적어도, VVC 후속 버전에 의해 생성된 비트 스트림의 고레벨 신택스 부분과 포워드 호환될 수 있도록 한다.

[0046] *VVC 제2 버전(VVCv2)에서의 일반 제약 정보*

[0047] VVC 제2 버전의 현재 초안(“VVC 동작 범위 확장(초안 5)”, ITU-T SG 16 WP 3 및 ISO/IEC JTC 1/SC 29 연합 비디오 전문가 그룹, JVET-X2005)에서, 일반 제약 플래그를 통해 제약한 일부 부가 코딩 툴을 제공한다. VVCv1에서 gc_num_reserved_bits로 지칭되는 8 비트의 필드를 gci_num_additional_bits로 개명하도록 제안한다. 제안된 VVCv2의 조정된 신택스는 아래와 같다.

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
...	
~~**gci_num_reserved_bits**~~	~~u(8)~~
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	

[0048]

for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
~~for(i = 0; i < gci_num_reserved_bits; i++)~~	
gci_reserved_zero_bit[i]	u(1)
}	
while(!byte_aligned( ))	
gci_alignment_zero_bit	f(1)
}	

[0049]

[0050] 부가 일반 제약 플래그는 모두 6 개, 즉 gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_rrc_rice_extension_constraint_flag, gci_no_persistent_rice_adaptation_constraint_flag 및 gci_no_reverse_last_sig_coeff_constraint_flag가 있다. gci_num_additional_bits 신택스 요소의 제안 해석(“시멘틱”)은 아래와 같다.

[0051] **gci_num_additional_bits**s는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits의 값은 0 또는 1과 같아야 한다. 1보다 큰 gci_num_additional_bits의 값은 ITU-T | ISO/IEC에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 또는 1보다 같아야 하는 것으로 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 1보다 큰 gci_num_additional_bits 값이 신택스에 나타나는 것을 허용해야 하고, gci_num_additional_bits가 1보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다.

[0052] VVCv2 비트 스트림이 일반 제약 정보를 전송하면, 6 개의 VVCv2 일반 제약 플래그가 전송되지 않은 경우, 신택스 요소 gci_num_additional_bits는 0으로 설정되어야 하고, 6 개의 VVCv2 일반 제약 플래그가 전송되는 경우, gci_num_additional_bits는 1로 설정되어야 하도록 제안한다.

[0053] 그러나, 위에서 제안한 VVCv2 일반 제약 신택스 규범은 VVCv1 디코더와 불호환되는 것을 초래한다. 구체적으로, 부가 일반 제약 플래그를 전송할 경우, 제안한 VVCv2 신택스는 gci_num_additional_bits를 1로 설정하는 것을 통해 상기 플래그를 전송한다. VVCv2 디코더에서, 일반 제약 정보를 전송할 경우, 비트 스트림으로부터 gci_num_additional_bits 신택스 요소를 8 비트의 무부호 정수로 디코딩한다. gci_num_additional_bits의 값이 1로 디코딩되면, 비트 스트림으로부터 6 개의 부가 비트를 디코딩한다. 상기 6 개의 비트를 VVCv2에서, 가능하게 제약되는 부가 디코딩 툴의 일반 제약 플래그로 해석한다.

[0054] VVCv1 디코더에서, 동일한 8 비트의 필드를 gci_num_reserved_bits로 해석한다. 그러나, 상기 신택스 요소의 값이 1로 디코딩되면, 비트 스트림으로부터 1 개의 부가 비트만 디코딩한다. 상기 비트는 버려지고, 사용되지 않는다. 따라서, 비트 스트림에 VVCv1 규범에 의해 식별되지 않는 5 개의 부가 비트가 존재할 수 있으므로, VVCv2 비트 스트림을 디코딩하고 있는 VVCv1 디코더는 비 동기화 문제에 직면할 수 있다.

[0055] 전술한 바와 같이, 비 동기화는 상이한 버전의 비디오 코딩 표준 사이의 엄중한 불호환 문제이다. VVCv2 비트 스트림은 VVCv2규범에서 이미 정의되었지만 VVCv1 디코더에서 모르고 있는 코딩 툴을 사용할 수 있으므로, VVCv1 디코더는 VVCv2 비트 스트림의 전부 콘텐츠를 디코딩할 수 없다. 그러나, VVCv1 디코더는 VVCv2 비트 스트림의 고레벨 신택스 부분을 적어도 디코딩할 수 있어야 하는 것이 바람직하다. 고레벨 신택스를 성공적으로 디코딩하는 것을 통해, 비디오 디코더는 일반 제약 정보를 결정할 수 있을 뿐만 아니라, 또한 프로파일 및 계층 정보를 결정할 수 있다. 이러한 정보는 디코더를 위해, 비트 스트림을 디코딩하는데 필요한 능력의 힌트를 제공한다. 예컨대, 일반 제약 정보는 디코더에 지시를 제공하여, 어느 코딩 툴이 비트 스트림의 제약을 받는지를 지시한다. 프로파일 및 계층 정보는 디코더에 지시를 제공하여, 지원을 획득해야 하는 미압축된 비디오 데이터 쓰

루풋(예컨대 비디오 데이터 레이트, 프레임 레이트, 해상도 등)을 지시한다.

[0056] 고레벨 신택스를 성공적으로 디코딩하면, 비디오 디코더는 현재 비트 스트림이 디코딩될 수 있는지 여부를 결정할 수 있고, 디코딩될 수 없으면, 디코더는 디코딩 과정을 여유롭게 중지할 수 있다. 반면, 고레벨 신택스 디코딩 과정에서의 비 동기화는 고레벨 신택스에서 제공된 정보가 정확하게 디코딩될 수 없는 것을 의미한다. 최악의 경우, 디코더는 비 동기화가 발생한 후 완전히 잘못된 신택스 요소값을 디코딩할 수 있음으로써, 잘못된 파라미터 설정과, 후속 저레벨 신택스에 대한 잘못된 디코딩을 초래하므로, 디코딩이 실패된다.

[0057] 또한, VVCv2 디코더에서, 일반 제약 정보를 전송할 경우, 비트 스트림으로부터 gci_num_additional_bits 신택스 요소를 8 비트의 무부호 정수로 디코딩한다. gci_num_additional_bits의 값이 0으로 디코딩되면, 일반 제약 플래그를 추가로 전송하지 않는다. 이러한 경우, 부가 일반 제약 플래그를 위해 추론값을 규정하지 않았고, 그 값도 정의되지 않았다. 따라서, 부가 일반 제약 플래그와 관련된 코딩 툴이 제약되어야 하는지 제약되지 않아야 하는지의 행위가 불명확하므로, 디코더의 구현 방식의 불일치를 초래할 수 있다.

[0058] VVC 규범에서, 신택스 요소 gci_reserved_zero_bit[i]의 명칭은 이러한 신택스 요소의 값이 무조건 0이여야 함을 시사하는 오해의 소지가 있다. 일반적으로, 인코더가 플레이스홀더로서 보류 신택스 요소를 비트 스트림에 기입할 경우, 보류 신택스 요소의 명칭에 디폴트값을 내장한다. 그러나, 일반 제약 신택스 구조의 설계는 인코더가 gci_reserved_zero_bit[i]를 영원히 기입하지 않음을 의미한다. gci_reserved_zero_bit[i]는 다만 특정 버전의 VVC 디코더가 더욱 높은 버전의 VVC 비트 스트림을 판독할 경우 사용된다. 이러한 경우, gci_reserved_zero_bit[i]의 값이 0인 것을 보장할 수 없다. 아래에서 일부 해결 방안을 제공하여 상기 문제를 해결한다.

[0059] *일반 제약 정보 전송*

[0060] 하나의 실시예에서, 일반 제약 정보를 전송하여 상기 비 동기화 문제를 해결한다. VVCv2 비트 스트림이 일반 제약 정보를 전송하고(즉, gci_present_flag의 값이 1임), VVCv2 일반 제약 정보가 가능하게 제약되는 N 개의 부가 코딩 툴로 구성되면, 신택스 요소 gci_num_additional_bits의 값은 0 또는 N으로만 설정될 수 있다. gci_num_additional_bits가 0으로 설정되면, 상기 N 개의 부가 코딩 툴의 일반 제약 플래그를 전송하지 않는다. gci_num_additional_bits가 N으로 설정되면, 비트 스트림에서의 다음 N 개의 비트는 상기 N 개의 부가 코딩 툴의 일반 제약 플래그를 전송하는데 사용된다. 하나의 예에서, N은 6으로 설정된다.

[0061] VVCv2 비트 스트림은 gci_num_additional_bits를 0 또는 N 이외의 값으로 설정하는 것을 허용하지 않는다. 그러나, VVCv2 디코더는 gci_num_additional_bits를 0 또는 N 이외의 값으로 설정한 일반 제약 정보를 여전히 처리할 수 있다. 예컨대, gci_num_additional_bits의 값을 0보다 크고 N보다 작거나, N보다 큰 것으로 설정할 수 있다. gci_num_additional_bits의 디코딩값을 M으로 설정한다. M이 0보다 크지만 N보다 작으면(즉, 0<M<N), gci_num_additional_bits 신택스 요소를 디코딩한 후, 디코더는 비트 스트림으로부터 gci_reserved_zero_bit[i] 신택스 요소로서 M 개의 비트를 추출하고 버린다. M이 N보다 크면(즉, N<M), gci_num_additional_bits 신택스 요소를 디코딩한 후, 디코더는 비트 스트림으로부터 N 개의 비트를 추출하고, 상기 N 개의 비트를 N 개의 부가 코딩 툴의 일반 제약 플래그로 해석한다. 다음, 디코더는 다시 비트 스트림으로부터 gci_reserved_zero_bit[i] 신택스 요소로서 M-N 개의 비트를 추출하고 버린다.

[0062] 다른 예에서, VVCv2 디코더는 gci_num_additional_bits를 0보다 크지만 N보다 작은 값을 설정한 일반 제약 정보를 처리할 필요가 없다. 합법적인 VVCv1비트 스트림은 gci_num_reserved_bits의 값을 0으로만 설정할 수 있다. 합법적인 VVCv2 비트 스트림은 다만 gci_num_reserved_bits의 값을 0 또는 N으로만 설정할 수 있다. VVC 미래 버전의 비트 스트림은 gci_num_additional_bits을 0부터 N 사이의 값으로 설정하는 것을 허용하지 않는다.

[0063] 본 실시예의 하나의 예에서, VVCv2의 일반 제약 정보 신택스를 VVCv2 코딩 툴을 위해 제출한 6 개의 일반 제약 플래그로 수정하고, 상기 수정은 아래의 표 1에 도시된 바와 같으며(추가된 부분은 밑줄로 나타내고, 삭제된 부분은 삭제선으로 나타냄), 여기서 “if(gci_num_additional_bits>0)”는 “if(gci_num_additional_bits>5)”로 대체된다.

[0064] 표 1

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)

[0065]

if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0 5) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
gci_reserved_zero_bit[i]	u(1)
}	
while(!byte_aligned( ))	
gci_alignment_zero_bit	f(1)
}	

[0066]

[0067] 하나의 예에서, VVCv2에서의 코딩 툴이 6 개의 부가 일반 제약 플래그를 구비하면, gci_num_additional_bits의 상응한 시멘틱은 아래와 같다(추가된 부분은 밑줄로 나타내고, 삭제된 부분은 삭제선으로 나타냄).

gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 ~~16~~ 과 같아야 한다. ~~1보다 큰~~ 0 또는 6 이외의 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits 의 값이 0 또는 ~~16~~ 과 같아야 하는 것으로 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 ~~1보다 큰~~ 0 및 6 이외의 gci_num_additional_bits 값이 신택스에 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 ~~1보다 큰~~ 0 또는 6 이외일 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다.

[0068]

[0069] 전술한 gci_num_additional_bits 시멘틱의 예에서, 상기 0 또는 6 이외, gci_num_additional_bits는 또한 0 또는 6 이외의 값을 취하는 것을 허용해야 한다. 다시 말해서, gci_num_additional_bits의 값은 1과 5 사이에 있을 수 있다. gci_num_additional_bits의 값은 6보다 클 수도 있다. gci_num_additional_bits의 값(M으로 표기됨)이 1과 5 사이에 있으면, 상기 표 1에 도시된 신택스에 따라, 디코더는 “if” 조건이 참일 경우 실행되는 단계를 스킵하고, “else” 단계로 넘어가, “numAdditionalBitsUsed”의 값에 0을 부여한다. 다음, “for” 순환에서 M 개의 비트를 판독하고 버린다. 이로써, 비 동기화 문제를 피할 수 있다. gci_num_additional_bits의 값 M이 6보다 크면, 6 개의 부가 일반 제약 플래그를 추출하고, 나머지 M-6 개의 비트를 추가로 추출하여 버린다.

[0070] 다른 하나의 예에서, VVCv2에서의 코딩 툴이 6 개의 부가 일반 제약 플래그를 구비하면, gci_num_additional_bits의 상응한 시멘틱은 아래와 같다(추가된 부분은 밑줄로 나타내고, 삭제된 부분은 삭제선으로 나타냄).

gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 16 과 같아야 한다. 16 보다 큰 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 또는 16 과 같도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 16 보다 큰 gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 16 보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다.

[0071]

[0072] 일반 제약 정보를 전송하는 다른 하나의 실시예에서, VVCv2 비트 스트림이 일반 제약 정보를 전송하고(즉, gci_present_flag의 값이 1임), VVCv2 일반 제약 정보가 가능하게 제약되는 N 개의 부가 코딩 툴로 구성되면, 신택스 요소 gci_num_additional_bits은 값 M으로 설정될 수 있다. M의 범위는 0 내지 N(0 및 N을 포함함)(0≤M≤N)이다. gci_num_additional_bits가 0으로 설정되면, 상기 N 개의 부가 코딩 툴의 일반 제약 플래그를 전송하지 않는다. gci_num_additional_bits가 0이 아닌 값 M로 설정되면, 비트 스트림에서의 다음 M 개의 비트는 상기 N 개의 부가 코딩 툴에서의 M 개의 일반 제약 플래그를 전송하는데 사용된다. 부가 코딩 툴에서의 어느 M 개가 제약될지는 일반 제약 정보 신택스 표에서 일반 제약 플래그가 나타나는 순서를 통해 결정된다.

[0073] 본 실시예의 하나의 예에서, VVCv2의 일반 제약 정보 신택스를 VVCv2 코딩 툴을 위해 제출한 현재 6 개의 일반 제약 플래그로 수정하고, 상기 수정은 아래의 표에 도시된 바와 같다.

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0)	

[0074]

gci_all_rap_pictures_constraint_flag	u(1)
if(gci_num_additional_bits > 1)	
gci_no_extended_precision_processing_constraint_flag	u(1)
if(gci_num_additional_bits > 2)	
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
if(gci_num_additional_bits > 3)	
gci_no_rrc_rice_extension_constraint_flag	u(1)
if(gci_num_additional_bits > 4)	
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
if(gci_num_additional_bits > 5)	
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
if(gci_num_additional_bits > 6) {	
numAdditionalBitsUsed = 6	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
gci_reserved_zero_bit[i]	u(1)
}	
}	
while(!byte_aligned( ))	
gci_alignment_zero_bit	f(1)
}	

[0075]

[0076] 본 실시예의 다른 하나의 예에서, 더욱 컴팩트한 신택스 표를 통해 등가 행위를 구현할 수 있다.

[0077]

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0)	
gci_all_rap_pictures_constraint_flag	u(1)
if(gci_num_additional_bits > 1)	
gci_no_extended_precision_processing_constraint_flag	u(1)
if(gci_num_additional_bits > 2)	
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
if(gci_num_additional_bits > 3)	
gci_no_rrc_rice_extension_constraint_flag	u(1)
if(gci_num_additional_bits > 4)	
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
if(gci_num_additional_bits > 5)	
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)

for(i = 0; i < gci_num_additional_bits - 6; i++)	
gci_reserved_zero_bit[i]	u(1)
}	
}	
while(!byte_aligned( ))	
gci_aligment_zero_bit	f(1)
}	

[0078]

[0079] VVCv2 코딩 툴의 일반 제약 플래그의 순서를 개변하는 것을 통해, 일반 제약 정보 신택스에서 본 실시예의 대안적인 배치를 서술할 수 있다.

[0080] VVCv2에서의 코딩 툴이 6 개의 부가 일반 제약 플래그를 구비하면, gci_num_additional_bits의 상응한 시멘틱은 아래와 같을 수 있다.

[0081] **gci_num_additional_bits**는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits의 값은 0 내지 6(0 및 6을 포함함)이여야 한다. 6보다 큰 gci_num_additional_bits의 값은 ITU-T | ISO/IEC에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 내지 6(0 및 6을 포함함)이도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 6보다 큰 gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야 하고, gci_num_additional_bits가 6보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다.

[0082] 본 실시예에서, gci_num_additional_bits의 값에 따라, 일부 또는 전부 부가 일반 제약 플래그를 전송하지 않을 수 있다. 본 실시예의 하나의 배치에서, VVCv2 비트 스트림이 일반 제약 정보를 전송할 경우(즉, gci_present_flag플래그의 값이 1이면), 전송되지 않은 부가 일반 제약 플래그에 대응되는 코딩 툴에 제약을 가하지 않는다. 아래와 같이, gci_num_additional_bits의 시멘틱을 수정하는 것을 통해 이러한 행위를 서술할 수 있다.

[0083] **gci_num_additional_bits**는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits의 값은 0 내지 6(0 및 6을 포함함)이여야 한다. 6보다 큰 gci_num_additional_bits의 값은 ITU-T | ISO/IEC에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 내지 6(0 및 6을 포함함)이도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 6보다 큰 gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야 하고, gci_num_additional_bits가 6보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. 또한, gci_present_flag가 1과 같을 경우, general_constraints_info()의 신택스에 대응되는 제약 플래그가 존재하지 않는 코딩 툴에 제약을 가하지 않는다.

[0084] 본 실시예의 다른 하나의 배치에서, VVCv2 비트 스트림이 일반 제약 정보를 전송하고(즉, gci_present_flag의 값이 1이면), 부가 일반 제약 플래그가 전송되지 않은 경우, 상응한 툴은 제약된다. 아래와 같이, 부가 일반 제약 플래그의 시멘틱을 수정하는 것을 통해 이러한 행위를 서술할 수 있다.

[0085] **gci_num_additional_bits**는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits의 값은 0 내지 6(0 및 6을 포함함)이여야 한다. 6보다 큰 gci_num_additional_bits의 값은 ITU-T | ISO/IEC에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 내지 6(0 및 6을 포함함)이도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 6보다 큰

gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야 하고, gci_num_additional_bits가 6보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다.

[0086] **gci_all_rap_pictures_constraint_flag**가 1과 같으면 OlsInScope에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 규정한다. gci_all_rap_pictures_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_present_flag가 1과 같고 gci_all_rap_pictures_constraint_flag가 존재하지 않을 경우, gci_all_rap_pictures_constraint_flag의 값은 1과 같은 것으로 추론된다.

[0087] **gci_no_extended_precision_processing_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_extended_precision_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_extended_precision_processing_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_present_flag가 1과 같고 gci_no_extended_precision_processing_constraint_flag가 존재하지 않을 경우, gci_no_extended_precision_processing_constraint_flag의 값은 1과 같은 것으로 추론된다.

[0088] **gci_no_ts_residual_coding_rice_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_ts_residual_coding_rice_present_in_sh_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_ts_residual_coding_rice_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_present_flag가 1과 같고 gci_no_ts_residual_coding_rice_constraint_flag가 존재하지 않을 경우, gci_no_ts_residual_coding_rice_constraint_flag의 값은 1과 같은 것으로 추론된다.

[0089] **gci_no_rrc_rice_extension_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_rrc_rice_extension_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_rrc_rice_extension_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_present_flag가 1과 같고 gci_no_rrc_rice_extension_constraint_flag가 존재하지 않을 경우, gci_no_rrc_rice_extension_constraint_flag의 값은 1과 같은 것으로 추론된다.

[0090] **gci_no_persistent_rice_adaptation_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_persistent_rice_adaptation_enabled_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_persistent_rice_adaptation_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_present_flag가 1과 같고 gci_no_persistent_rice_adaptation_constraint_flag가 존재하지 않을 경우, gci_no_persistent_rice_adaptation_constraint_flag의 값은 1과 같은 것으로 추론된다.

[0091] **gci_no_reverse_last_sig_coeff_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_reverse_last_sig_coeff_enabled_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_reverse_last_sig_coeff_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_present_flag가 1과 같고 gci_no_reverse_last_sig_coeff_constraint_flag가 존재하지 않을 경우, gci_no_reverse_last_sig_coeff_constraint_flag의 값은 1과 같은 것으로 추론된다.

[0092] *일반 제약 정보 플래그 초기화*

[0093] 본 문은 일반 제약 정보 플래그를 초기화하는 하나의 실시예를 설명하여, 상기 디코더 구현 방식이 불명확하고 불일치한 문제를 해결한다. 본 실시예에서, gci_present_flag가 1과 같고 gci_num_additional_bits가 0과 같을 경우, general_constraints_info()는 gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag와 관련된 코딩 툴에 제약을 가하지 않는다.

[0094] 하나의 예로서, gci_num_additional_bits의 시멘틱의 가능한 변화는 아래와 같으며, 이러한 변화는 VVC 제2 버전 규범의 현재 버전에 기반하여 이루어진 것이다(추가된 부분은 밑줄로 나타내고, 삭제된 부분은 삭제선으로 나타냄).

[0095] **gci_num_additional_bits**는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits의 값은 0 또는 1과 같아야 한다. 1보다 큰 gci_num_additional_bits의 값은 ITU-T | ISO/IEC에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 또는 1보다 같아야 하는 것으로 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 1보다 큰 gci_num_additional_bits 값이 신택스에 나타나는 것을 허용해야 하고, gci_num_additional_bits가 1보다 클

경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. gci_present_flag가 1과 같고 gci_num_additional_bits가 0과 같을 경우, general_constraints_info()는 gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag와 관련된 코딩 툴에 제약을 가하지 않는다.

[0096] 다른 하나의 예로서, gci_num_additional_bits의 시멘틱의 가능한 변화는 아래와 같으며, 이러한 변화는 VVC 제2 버전 규범의 현재 버전에 기반하여 이루어진 것이다.

gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 ~~1~~16 과 같아야 한다. ~~1보다 큰~~ 0 또는 6 이외의 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits 의 값이 0 또는 ~~1~~16 과 같아야 하는 것으로 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 ~~1보다 큰~~ 0 및 6 이외의 gci_num_additional_bits 값이 신택스에 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 ~~1보다 큰~~ 0 또는 6 이외일 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. gci_present_flag 가 1 과 같고 gci_num_additional_bits 가 0 과 같을 경우, general_constraints_info()는 gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag 와 관련된 코딩 툴에 제약을 가하지 않는다.

[0097]

[0098] VVC 제2 버전 규범의 현재 버전에서, gci_num_additional_bits의 시멘틱은 다만 gci_present_flag가 1인 경우 사용된다. gci_present_flag가 0과 같은 시나리오는 VVC 제2 버전 규범의 다른 섹션에서 해결된다. 따라서, 상기 gci_num_additional_bits 시멘틱에서의 “gci_present_flag가 1과 같을 경우”는 자동적으로 만족한다. 또한, VVC 제2 버전 규범은 gci_num_additional_bits의 값이 1과 5 사이에 있는 것을 허용하지 않으므로, “gci_num_additional_bits가 0과 같은 것”은 “gci_num_additional_bits가 5보다 작거나 같은 것” 또는 “gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag,

gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag가 존재하지 않는 것"과 동일하다. 따라서, gci_num_additional_bits 시멘틱에 대한 상기 수정은 아래의 콘텐츠와 같다.

[0099] **gci_num_additional_bits** 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 ~~1~~16 과 같아야 한다. ~~1보다 큰~~ 0 또는 6 이외의 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits 의 값이 0 또는 ~~1~~16 과 같아야 하는 것으로 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 ~~1보다 큰~~ 0 및 6 이외의 gci_num_additional_bits 값이 신택스에 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 ~~1보다 큰~~ 0 또는 6 이외일 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag 가 존재하지 않을 경우, general_constraints_info()는 gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag 와 관련된 코딩 툴에 제약을 가하지 않는다.

[0100]

[0101] 아래의 몇 개의 예에서, 부가 일반 제약 플래그의 추론값은 상기 시멘틱과 유사한 방식으로 설정될 수 있다. 예컨대, 아래와 같이 부가 일반 제약 플래그의 시멘틱을 수정하여, 추론값 설정을 포함할 수 있도록 할 수 있다.

[0102] **gci_all_rap_pictures_constraint_flag**가 1과 같으면 OlsInScope에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 규정한다. gci_all_rap_pictures_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. 존재하지 않을 경우, gci_all_rap_pictures_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0103] **gci_no_extended_precision_processing_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_extended_precision_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_extended_precision_processing_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. 존재하지 않을 경우, gci_no_extended_precision_processing_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0104] **gci_no_ts_residual_coding_rice_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_ts_residual_coding_rice_present_in_sh_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_ts_residual_coding_rice_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. 존재하지 않을 경우, gci_no_ts_residual_coding_rice_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0105] **gci_no_rrc_rice_extension_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_rrc_rice_extension_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_rrc_rice_extension_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. 존재하지 않을 경우, gci_no_rrc_rice_extension_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0106] **gci_no_persistent_rice_adaptation_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_persistent_rice_adaptation_enabled_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_persistent_rice_adaptation_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. 존재하지 않을 경우, gci_no_persistent_rice_adaptation_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0107] **gci_no_reverse_last_sig_coeff_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_reverse_last_sig_coeff_enabled_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_reverse_last_sig_coeff_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. 존재하지 않을 경우, gci_no_reverse_last_sig_coeff_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0108] 다른 하나의 예에서, 부가 일반 제약 플래그의 시멘틱 수정은 아래와 같다.

[0109] **gci_all_rap_pictures_constraint_flag**가 1과 같으면 OlsInScope에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 규정한다. gci_all_rap_pictures_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_all_rap_pictures_constraint_flag가 존재하지 않을 경우, gci_all_rap_pictures_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0110] **gci_no_extended_precision_processing_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_extended_precision_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_extended_precision_processing_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_extended_precision_processing_constraint_flag가 존재하지 않을 경우, gci_no_extended_precision_processing_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0111] **gci_no_ts_residual_coding_rice_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_ts_residual_coding_rice_present_in_sh_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_ts_residual_coding_rice_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_ts_residual_coding_rice_constraint_flag가 존재하지 않을 경우, gci_no_ts_residual_coding_rice_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0112] **gci_no_rrc_rice_extension_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_rrc_rice_extension_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_rrc_rice_extension_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_rrc_rice_extension_constraint_flag가 존재하지 않을 경우, gci_no_rrc_rice_extension_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0113] **gci_no_persistent_rice_adaptation_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_persistent_rice_adaptation_enabled_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_persistent_rice_adaptation_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_persistent_rice_adaptation_constraint_flag가 존재하지 않을 경우, gci_no_persistent_rice_adaptation_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0114] **gci_no_reverse_last_sig_coeff_constraint_flag**가 1과 같으면 OlsInScope 중 모든 픽처의 sps_reverse_last_sig_coeff_enabled_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_reverse_last_sig_coeff_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_reverse_last_sig_coeff_constraint_flag가 존재하지 않을 경우, gci_no_reverse_last_sig_coeff_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0115] 다른 하나의 예에서, gci_num_additional_bits의 시멘틱의 수정은 아래와 같다.

gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 ±6 과 같아야 한다. ±6 보다 큰 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 또는 ±6 과 같도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 ±6 보다 큰 gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 ±6 보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다.

[0116]

gci_num_additional_bits 가 0 과 같을 경우, 부가 GCI 비트에 의해 규정된 모든 제약 플래그는 0과 같은 것으로 추론된다.

[0117]

[0118] 다른 하나의 예에서, gci_num_additional_bits의 시멘틱의 수정은 아래와 같다.

gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 ~~16~~ 과 같아야 한다. ~~16~~ 보다 큰 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 또는 ~~16~~ 과 같도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 ~~16~~ 보다 큰 gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 ~~16~~ 보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. 존재하지 않을 경우, 부가 GCI 비트에 의해 규정된 모든 제약 플래그는 0과 같은 것으로 추론된다.

[0119]

[0120] 다른 하나의 예에서, 일반 제약 정보 신택스 요소의 신택스 및 시멘틱 수정은 아래와 같다.

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0 5) {	
additional_general_constraints_info( )	
~~**gci_all_rap_pictures_constraint_flag**~~	~~u(1)~~
~~**gci_no_extended_precision_processing_constraint_flag**~~	~~u(1)~~
~~**gci_no_ts_residual_coding_rice_constraint_flag**~~	~~u(1)~~
~~**gci_no_rrc_rice_extension_constraint_flag**~~	~~u(1)~~
~~**gci_no_persistent_rice_adaptation_constraint_flag**~~	~~u(1)~~
~~**gci_no_reverse_last_sig_coeff_constraint_flag**~~	~~u(1)~~
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	

[0121]

gci_reserved_zero_bit[i]	u(1)
}	
while(!byte_aligned( ))	
gci_alignment_zero_bit	f(1)
}	

additional_general_constraints_info( ) {	디스크립터
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
}	

[0122] gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 ± 16 과 같아야 한다. ± 16 보다 큰 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 또는 ± 16 과 같도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 ± 16 보다 큰 gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야

[0123]

하고, gci_num_additional_bits 가 16 보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. gci_num_additional_bits 가 0 과 같을 경우, additional_general_constraints_info() 신택스 구조는 임의의 제약을 가하지 않는다.

[0124]

[0125] 다른 하나의 예에서, 일반 제약 정보 신택스 요소의 시멘틱 수정은 아래와 같다.

gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 16 과 같아야 한다. 16 보다 큰 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits 의 값이 0 또는 16 과 같도록 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 16 보다 큰 gci_num_additional_bits 값이 신택스에서 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 16 보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. 존재하지 않을 경우, additional_general_constraints_info() 신택스 구조는 임의의 제약을 가하지 않는다.

[0126]

[0127] 일반 제약 정보 플래그를 초기화하는 다른 하나의 실시예에서, gci_present_flag가 1과 같고 gci_num_additional_bits가 0과 같을 경우, gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag는 이러한 플래그 각각의 시멘틱에 의해 규정된 상응한 제약을 항상 가할 수 있다. 다시 말해서, GCI 플래그가 존재하고 gci_num_additional_bits가 0과 같을 경우, 상기 6 개의 GCI 플래그는 1과 같은 것으로 추론된다.

[0128] 하나의 예로서, gci_num_additional_bits의 시멘틱의 가능한 변화는 아래와 같으며, 이러한 변화는 VVC 제2 버전 규범의 현재 새로 증가된 콘텐츠에 기반하여 이루어진 것이다.

[0129] **gci_num_additional_bits**는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits의 값은 0 또는 1과 같아야 한다. 1보다 큰 gci_num_additional_bits의 값은 ITU-T |

ISO/IEC에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits의 값이 0 또는 1보다 같아야 하는 것으로 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 1보다 큰 gci_num_additional_bits 값이 신택스에 나타나는 것을 허용해야 하고, gci_num_additional_bits가 1보다 클 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. gci_present_flag가 1과 같고 gci_num_additional_bits가 0과 같을 경우, gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag는 1과 같은 것으로 추론된다.

[0130] 다른 하나의 예로서, gci_num_additional_bits의 시멘틱의 가능한 변화는 아래와 같으며, 이러한 변화는 VVC 제2 버전 규범의 현재 새로 증가된 콘텐츠에 기반하여 이루어진 것이다.

gci_num_additional_bits 는 일반 제약 정보 신택스 구조 중 gci_alignment_zero_bit 신택스 요소(존재할 경우) 이외의 부가 GCI 비트수를 규정한다. 본 문서의 상기 버전에 부합되는 비트 스트림에서, gci_num_additional_bits 의 값은 0 또는 ~~1~~6 과 같아야 한다. ~~1보다 큰~~ 0 또는 6 이외의 gci_num_additional_bits 의 값은 ITU-T | ISO/IEC 에 의해 미래에 사용되도록 보류된다. 본 문서의 상기 버전에서 gci_num_additional_bits 의 값이 0 또는 ~~1~~6 과 같아야 하는

[0131]

것으로 요구되지만, 본 문서의 상기 버전에 부합되는 디코더는 ~~1보다 큰~~ 0 및 6 이외의 gci_num_additional_bits 값이 신택스에 나타나는 것을 허용해야 하고, gci_num_additional_bits 가 ~~1보다 큰~~ 0 또는 6 이외일 경우 모든 gci_reserved_zero_bit[i] 신택스 요소의 값을 무시해야 한다. gci_present_flag 가 1 과 같고 gci_num_additional_bits 가 0 과 같을 경우, gci_all_rap_pictures_constraint_flag, gci_no_extended_precision_processing_constraint_flag, gci_no_ts_residual_coding_rice_constraint_flag, gci_no_reverse_last_sig_coeff_constraint_flag, gci_no_rrc_rice_extension_constraint_flag 및 gci_no_persistent_rice_adaptation_constraint_flag 는 1 과 같은 것으로 추론된다.

[0132]

[0133] 상기 실시예의 보충 또는 대안적인 것으로서, 신택스 요소에 의해 오도된 gci_reserved_zero_bit[i]를 gci_reserved_bit[i]로 재명명할 수 있다. 예컨대, 수정된 신택스 및 시멘틱은 아래와 같을 수 있다.

[0134]

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0 5) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
~~gci_reserved_zero_bit[i]~~ gci_reserved_bit[i]	u(1)
}	
while(!byte_aligned( ))	
gci_alignment_zero_bit	f(1)
}	

[0135]

[0136] ~~gci_reserved_zero_bit[i]~~ gci_reserved_bit[i] 는 임의의 값을 구비할 수 있다. gci_reserved_bit[i]의 존재 및 값은 본 규범의 상기 버전에 의해 규정된 디코딩 과정에 영향을 미치지 않는다. 본 규범의 상기 버전에 부합되는 디코더는 모든 ~~gci_reserved_zero_bit[i]~~gci_reserved_bit[i] 신택스 요소의 값을 무시해야 한다.

[0137]

[0138] 다른 하나의 실시예에서, 신택스 요소 gci_reserved_zero_bit[i]를 gci_reserved_bit[i]로 재명명한다. 예컨대, 수정된 신택스 및 시멘틱은 아래와 같을 수 있다.

[0139]

general_constraints_info( ) {	디스크립터
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 일반(general) */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0 5) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)

gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
~~gci_reserved_zero_bit[i]~~ gci_reserved_bit[i]	u(1)
}	
while(!byte_aligned( ))	
gci_alignment_zero_bit	f(1)
}	

[0140] ~~gci_reserved_zero_bit[i]~~ gci_reserved_bit[i] 는 임의의 값을 구비할 수 있다. gci_reserved_bit[i]의 존재 및 값은 본 규범의 상기 버전에 의해 규정된 디코딩 과정에 영향을 미치지 않는다. 본 규범의 상기 버전에 부합되는 디코더는 모든 ~~gci_reserved_zero_bit[i]~~gci_reserved_bit[i] 신택스 요소의 값을 무시해야 한다.

[0141]

[0142] *일반 제약 플래그*

[0143] **gci_all_rap_pictures_constraint_flag**

[0144] gci_all_rap_pictures_constraint_flag는 픽처를 IRAP 또는 GDR 픽처로 제한하는 것을 지시하기 위한 것이다.

[0145] 네트워크 추상화 레이어(NAL)는 VVC 신택스 요소를 “NAL 유닛”의 시스템 인터페이스로 구성한다. 이러한 구조는 VVC에 대해 간단하고 유효한 커스텀을 수행하는 것을 허용하여, 실시간 통신 애플리케이션으로부터 애플리케이션을 저장하기 위한 파일 포맷까지 등 광범위한 다양한 사용 예에 적용된다. VVC 표준에서 NAL 유닛 타입의 완전한 리스트는 아래의 표에 도시된 바와 같다.

nal_unit_type	Nal_unit_type 명칭	NAL 유닛 콘텐츠 및 RBSP 신택스 구조	NAL 유닛 타입 분류
0	TRAIL_NUT	트레일링 픽처 또는 서브 픽처의 코딩 슬라이스* slice_layer_rbsp( )	VCL
1	STSA_NUT	STSA 픽처 또는 서브 픽처의 코딩 슬라이스* slice_layer_rbsp( )	VCL
2	RADL_NUT	Coded slice of a RADL picture or subpicture* slice_layer_rbsp( )	VCL
3	RASL_NUT	RASL 픽처 또는 서브 픽처의 코딩 슬라이스* slice_layer_rbsp( )	VCL
4..6	RSV_VCL_4..	보류된 비 IRAP VCL NAL 유닛 타입	VCL

[0146]

	RSV_VCL_6		
7 8	IDR_W_RADL IDR_N_LP	IDR 픽처 또는 서브 픽처의 코딩 슬라이스* slice_layer_rbsp( )	VCL
9	CRA_NUT	CRA 픽처 또는 서브 픽처의 코딩 슬라이스* slice_layer_rbsp( )	VCL
10	GDR_NUT	GDR 픽처 또는 서브 픽처의 코딩 슬라이스* slice_layer_rbsp( )	VCL
11	RSV_IRAP_11	보류된 IRAP VCL NAL 유닛 타입	VCL
12	OPI_NUT	동작 포인트 정보 operating_point_information_rbsp( )	비-VCL
13	DCI_NUT	디코딩 능력 정보 decoding_capability_information_rbsp( )	비-VCL
14	VPS_NUT	비디오 파라미터 세트 video_parameter_set_rbsp( )	비-VCL
15	SPS_NUT	시퀀스 파라미터 세트 seq_parameter_set_rbsp( )	비-VCL
16	PPS_NUT	픽처 파라미터 세트	비-VCL

[0147]

		pic_parameter_set_rbsp( )	
17 18	PREFIX_APS_NUT SUFFIX_APS_NUT	적응 파라미터 세트 adaptation_parameter_set_rbsp( )	비-VCL
19	PH_NUT	픽처 헤드 picture_header_rbsp( )	비-VCL
20	AUD_NUT	AU 딜리미터 access_unit_delimiter_rbsp( )	비-VCL
21	EOS_NUT	시퀀스 종료 end_of_seq_rbsp( )	비-VCL
22	EOB_NUT	비트 스트림 종료 end_of_bitstream_rbsp( )	비-VCL
23 24	PREFIX_SEI_NUT SUFFIX_SEI_NUT	보충 강화 정보 sei_rbsp( )	비-VCL
25	FD_NUT	충진 데이터 filler_data_rbsp( )	비-VCL
26 27	RSV_NVCL_26 RSV_NVCL_27	보류된 비-VCL NAL 유닛 타입	비-VCL
28..31	UNSPEC_28..	미규정된 VCL NAL 유닛 타입	비-VCL

[0148]

	UNSPEC_31		
* pps_mixed_nalu_types_in_pic_flag 가 0 과 같을 경우, 픽처 속성을 지시하고; pps_mixed_nalu_types_in_pic_flag 가 1 과 같을 경우, 서브 픽처 속성을 지시한다.			

[0149]

[0150] 비디오 코딩층(VCL)으로 분류된 NAL 유닛은 저레벨 신택스 요소를 포함하고, 비-VCL로 분류된 NAL 유닛은 고레벨 신택스 요소를 포함한다. 비디오 시퀀스의 픽처는 VCL NAL 유닛으로부터 디코딩된다. 상이한 타입의 VCL 카테고리는 하이 레벨에서 의존 관계를 지시하는데 사용될 수 있다. 예컨대, TRAIL_NUT (0)으로부터 RSV_VCL_6

(6)까지의 NAL 유닛은 일반적으로 인터 예측 툴을 사용할 수 있으며, 이는 이전의 디코딩 (참조) 픽처에 액세스할 수 있는지 여부에 따라 결정된다. 인트라 예측 툴만 사용하여 압축된 픽처에 비해, 인터 예측 툴을 사용하여 코딩된 픽처의 압축 효율이 더욱 높다. 그러나, 참조 픽처를 획득할 수 없는 경우, 이러한 디코딩 의존성은 문제를 이르킨다.

[0151] 인트라 랜덤 액세스 포인트(IRAP) 픽처는 코딩 픽처로서, 모든 VCL NAL 유닛의 nal_unit_type 값이 동일하고, 모두 IDR_W_RADL 내지 CRA_NUT(IDR_W_RADL 및 CRA_NUT를 포함함)의 범위 내에 있다. IRAP 픽처는 CRA 픽처일 수 있고, IDR 픽처일 수도 있다. IRAP 픽처는 디코딩 과정에서 동일한 층의 참조 픽처로부터의 인터 예측을 사용하지 않는다. 비트 스트림 중 디코딩 순서에 따라 배열된 첫 번째 픽처는 IRAP 픽처 또는 점진적 디코딩 리프레시(GDR) 픽처이다. 단일 층 비트 스트림의 경우, 참조해야 할 경우 필요한 파라미터 세트를 획득할 수만 있으면, 디코딩 순서가 IRAP 픽처 이전인 임의의 픽처에 대해 디코딩 처리를 실행할 필요가 없이, IRAP 픽처 및 CLVS 중 디코 딩순서에 따라 배열된 모든 후속 비 RASL 픽처를 정확하게 디코딩할 수 있다. IRAP 픽처의 pps_mixed_nalu_types_in_pic_flag 값은 0과 동일하다. 하나의 픽처의 pps_mixed_nalu_types_in_pic_flag가 0과 같고, 상기 픽처의 임의의 슬라이스의 nal_unit_type가 IDR_W_RADL 내지 CRA_NUT(IDR_W_RADL 및 CRA_NUT를 포함함)의 범위 내에 있으면, 상기 픽처의 모든 다른 슬라이스의 nal_unit_type값은 동일하고, 상기 픽처는 첫 번째 슬라이스를 수신한 후 IRAP 픽처인 것이 알려진다.

[0152] 따라서, IRAP 픽처는 동일한 층을 가로 지르는 인터 예측을 사용하지 않는다. 이러한 제한은 IRAP 픽처가 스트리밍 비디오 애플리케이션의 에러 복구 포인트로 되거나, 주문형 비디오 재생 애플리케이션의 검색 위치로 되는 것을 허용한다. 그러나, IRAP 픽처의 압축 효율은 일반적으로 비 IRAP 픽처보다 낮다.

[0153] VVC 표준에 점진적 디코딩 리프레시(GDR) 픽처를 인입하여, 비 IRAP와 IRAP 픽처 사이의 트레이드오프로 사용한다. GDR 픽처는 인터 예측을 사용하지 않는 일부 “클린” 부분을 구비하고, 픽처의 나머지 부분은 인터 예측을 자유로 사용할 수 있다. 이러한 방식을 통해 픽처를 분할하면, 데이터 패킷 손실 등 에러 이벤트가 발생할 겨우, “클린” 부분이 여전히 정확하게 디코딩될 수 있다. 연속적인 GDR 픽처에서, “클린” 부분의 공간 위치는 회전되므로, 최종적으로 에러로부터 전체 픽처를 복구할 수 있다.

[0154] 재생 유연한 스트리밍 복원 능력이 매우 중요한 비디오 애플리케이션의 경우, 모든 픽처를 IRAP 또는 GDR 픽처로 제한하는 것이 바람직할 수 있다. VVCv2에서, GCI 플래그 gci_all_rap_pictures_constraint_flag를 인입하여, 하이 레벨에서 이러한 제한을 지시할 수 있다. gci_all_rap_pictures_constraint_flag가 1과 같으면 OlsInScope에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 규정한다. gci_all_rap_pictures_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_all_rap_pictures_constraint_flag가 존재하지 않을 경우, gci_all_rap_pictures_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0155] profile_tier_level() 신택스 구조가 VPS에 포함될 경우, OlsInScope는 VPS에 의해 규정된 하나 또는 복수 개의 출력층 세트(OLS)이다. profile_tier_level() 신택스 구조가 SPS에 포함될 경우, OlsInScope는 SPS를 인용한 층에서의 최저층만 포함하는 OLS이고, 상기 최저층은 독립층이다.

[0156] **gci_no_extended_precision_processing_constraint_flag**

[0157] GCI 플래그 gci_no_extended_precision_processing_constraint_flag는 하이 레벨에서 확장 변환 정밀도의 VVCv2 툴이 제약되는지를 전송한다. VVC 표준에서, 비디오 신호의 픽셀을 정수값으로 나탄낸다. VVC 표준에서 설명된 모든 계산 및 처리는 모두 정수에 대한 연산을 통해 나타낸 것이다. 복잡성 및 상호 운용성 이유로, 이러한 제한이 매우 중요하다. 먼저, 정수 연산(덧셈, 곱셈, 나눗셈)의 계산 비용은 일반적으로 등가 부동 소수점 연산보다 낮다. 다음, 부동 소수점 연산은 결정적으로 표준화되지 않았다. 부동 소수점 덧셈 및 나눗셈은 무조건 교환적인 것이 아니며(예컨대, (a+b)+c는 a+(b+c)와 무조건 같은 것이 아님), 상이한 플랫폼에서의 부동 소수점 연산의 평가가 완전히 동일한 것도 보장할 수 없다.

[0158] 비디오 신호 샘플의 비트 깊이는 비디오 소스의 속성이고, VVC 표준에서 BitDepth으로 라벨링된다. 혼합 비디오 코딩 시스템에서, 인터 예측 또는 인트라 예측 툴을 통해 비디오 샘플을 예측한다. 원시 비디오 샘플과 예측 샘플 사이의 차이값을 잔차로 지칭한다. 최악의 경우, 이러한 잔차 계수의 비트 깊이는 확대될 수 있지만(예컨대, BitDepth+1); VVC 표준에서, 이러한 잔차 계수는 클리핑되어 비트 깊이 BitDepth가 유지된다. 실제로, 실제의 인코더는 원시 비디오 신호보다 폭이 더욱 큰 잔차를 생성하는 예측 툴을 선택하지 않으므로, 최악의 경우가 발생하지 않는다.

[0159] 다음, 일반적으로 이산 코사인 변환(DCT)을 정수화하는 것을 통해 잔차 계수를 분석하여, 변환 계수를 생성한다. 이산 코사인 변환(DCT)은 선형이고, 가역적인 함수이며, 그 형식은 아래와 같이 나타낼 수 있다.

[0160] $f: \mathbb{R}^N \rightarrow \mathbb{R}^N$

[0161] 여기서, $\mathbb{R}^N$은 실수의 N 차원 공간을 나타낸다. VVC 표준에서 DCT의 정수화 근사를 사용한다(즉, $f: \mathbb{Z}^N \rightarrow \mathbb{Z}^N$). 예측과 상이한 것은, 변환된 정밀도는 코딩 이득에 영향을 미치므로, 변환 스테이지에서 비트 깊이에 대해 일정한 확장을 수행하는 것을 일반적으로 허용한다. 근사한 DCT 계수의 비트 깊이와, 결과로 사용되는 변환 계수의 비트 깊이는 하드웨어 복잡성과 코딩 성능 사이에서 트레이드 오프를 수행하는 설계 결정이다.

[0162] VVCv1에서, 변환 계수의 비트 깊이는 16이다. 다시 말해서, 각 변환 계수의 값 범위는 $[-2^{15}, 2^{15}-1]$이다. 비디오 샘플과 정수화된 DCT 계수를 서로 곱하면, 일반적으로 비트 깊이가 16보다 큰 중간 변환 계수가 생성된다. 필요한 비트 깊이를 구비하는 변환 계수를 생성하기 위해, 중간 변환 계수를 왼쪽으로 비트 시프트한다. 이러한 동작 자체는 손실이 있는 것이다.

[0163] VVCv2에서, SPS플래그 sps_extended_precision_flag를 1로 설정하는 것을 통해 확장 변환 정밀도를 인에이블할 수 있다. 확장 변환 정밀도가 인에이블되면, 변환 계수비트 깊이는 (Log2TransformRange+1)로 증가된다. 정확한 변환 계수의 비트 깊이는 BitDepth에 의해 결정된다. sps_extended_precision_flag가 1과 같으면 스케일링 및 변환 과정에서 확장 동적 범위를 변환 계수에 사용하고, 확장 동적 범위를 abs_remaining[ ] 및 dec_abs_level[ ] 신택스 요소의 이진화에 사용하는 것으로 규정한다. sps_extended_precision_flag가 0과 같으면 스케일링 및 변환 과정에서 확장 동적 범위를 사용하지 않고, 확장 동적 범위를 abs_remaining[ ] 및 dec_abs_level[ ] 신택스 요소의 이진화에 사용하지 않는 것으로 규정한다. 존재하지 않을 경우, sps_extended_precision_flag의 값은 0과 같은 것으로 추론된다. 변수 Log2TransformRange의 유도 과정은 아래와 같다.

[0164] Log2TransformRange = sps_extended_precision_flag ? Max(15, Min(20, BitDepth + 6)) : 15

[0165] CoeffMin = -(1 << (sps_extended_precision_flag ? Max(15, Min(20, BitDepth + 6)) : 15))

[0166] CoeffMax = (1 << (sps_extended_precision_flag ? Max(15, Min(20, BitDepth + 6)) : 15)) - 1

[0167] gci_no_extended_precision_processing_constraint_flag가 1과 같으면 OlsInScope 중 모든 픽처의 sps_extended_precision_flag가 모두 0과 같아야 하는 것으로 규정한다. 본 문에서 OlsInScope는 또한 “범위 내 출력층 세트”로 지칭된다. profile_tier_level() 신택스 구조가 VPS에 포함될 경우, OlsInScope는 VPS에 의해 규정된 하나 또는 복수 개의 출력층 세트(OLS)이다. profile_tier_level() 신택스 구조가 SPS에 포함될 경우, OlsInScope는 SPS를 인용한 층에서의 최저층만 포함하는 OLS이고, 상기 최저층은 독립층이다. gci_no_extended_precision_processing_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_extended_precision_processing_constraint_flag가 존재하지 않을 경우, gci_no_extended_precision_processing_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0168] **gci_no_ts_residual_coding_rice_constraint_flag**

[0169] GCI 플래그 **gci_no_ts_residual_coding_rice_constraint_flag**는 하이 레벨에서 명시적 라이스 파라미터 전송의 VVCv2 툴이 제약되는지 여부를 전송한다. 엔트로피 코딩하는 경우, 엔트로피 코딩 과정은 각 신택스 요소값을 하나의 비트 시퀀스로 코딩하고, 비트 스트림에 삽입한다.

[0170] VVC에서 사용되는 두 개의 엔트로피 코딩 과정은 컨텍스트 적응 이진 산술 코딩(CABAC) 및 라이스 코딩이다. CABAC 엔진은 적응적인 것으로서, 신택스 요소를 이론적 섀넌 한계에 매우 접근하는 비트레이트로 압축할 수 있다. 그러나, 산술 코딩은 매우 복잡하다. CABAC를 사용하여 비 이진 신택스 요소(예컨대, 값 범위가 0 또는 1을 초과한 신택스 요소)를 코딩하기 위해, 먼저 신택스 요소를 한 그룹의 “빈”으로 이진화한다. 아래의 표에서 두 개의 이진화의 예를 제공한다. 두 번째 예는, 가변 길이 이진화의 경우, 신택스 요소의 특정된 값에 빈이 가능하게 존재하지 않는 것을 보여준다.

[0171] 표: 고정 길이 이진화 예

값	이진화
0	00
1	01
2	10
3	11

[0172]

[0173] 표: 가변 길이 이진화 예

값	이진화
0	0
1	10
2	110
3	1110
4	1111

[0174]

[0175] 이진화에서의 각 빈은 CABAC 엔진에 의해 코딩될 수 있다. 그러나, CABAC를 사용하여 빈을 코딩하기 위해, 엔진은 관련된 "콘텐츠"를 무조건 저장하고 업데이트해야 한다. 상기 콘텐츠는 빈의 확률 분포를 모델링한다. 신택스 요소의 이진화에서, 각 빈은 모두 단독적인 콘텐츠가 필요하다.

[0176] 값 범위가 비교적 큰 신택스 요소는 단순히 CABAC 를 사용하여 코딩되는 것이 적합하지 않으며, 그 원인은 이러한 타입의 신택스 요소의 이진화가 비교적 길므로, 콘텐츠 저장 및 업데이트 측면에서 과도한 오버헤드를 야기하기 때문이다. 예컨대, 잔차 계수값은 단순히 CABAC를 사용하여 코딩되는 것이 바람직하지 않다. CABAC에 비해, 라이스 코딩은 컴팩트한 파라미터로 비 이진 값의 확률 분포를 모델링할 수 있다.

[0177] 라이스 코딩은 골롬 코딩 또는 라이스-골롬 코딩으로 지칭될 수도 있다. 골롬 코딩은 단일 파라미터 M에 의해 제어되는 엔트로피 인코더이고, 상기 파라미터 M은 양의 정수로 제한된다. 라이스 코딩은 골롬 코딩의 하나의 서브 세트이고, 여기서, 엔트로피 인코더는 단일 라이스 파라미터 R에 의해 제어되며, R은 음이 아닌 정수이다. 라이스 파라미터 R을 구비하는 라이스 코딩은 골롬 코딩에 등가되고, 여기서 $M = 2^R$이다.

[0178] 아래와 같이, 음이 아닌 정수값 x를 라이스 파라미터 R을 구비하는 라이스 코딩으로 이진화한다. 몫 q 및 여수 r의 계산 공식은 아래와 같다.

[0179] $$q = \text{floor}\left(\frac{x}{2^R}\right)$$

[0180] $$r = x \text{ modulo } 2^R$$

[0181] x의 라이스 코딩은 프리픽스 코딩 및 서픽스 코딩의 조합이다. 프리픽스 코드는 q의 단항 코딩에 의해 결정된다. 예컨대, VVC에서 사용되는 프리픽스 코딩은 절단된 단항 코딩이다.

q	단항 코딩
0	0
1	10
2	110
3	1110
4	11110
5	111110
6	111111

[0182]

[0183] 단항 코딩이 절단되었으므로, 라이스 코딩도 절단되며, 이는 코딩된 x의 값 범위가 제한된 것을 의미한다. VVC에서, 트렁게이트드 라이스 코딩은 범위가 $[0,6*2^R]$인 x값에 적용된다.

[0184] 서픽스 코딩은 길이가 R 개의 비트인 r의 고정 길이 코딩의 이진화이다. 예컨대, R=3이면, 서픽스 코드는 아래와 같다.

r	고정 길이 코딩
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

[0185]

[0186] VVC에서, CABAC, 라이스 코딩 및 지수 골롬 코딩의 조합을 통해 잔차 계수를 코딩한다. 적은 양의 신택스 요소 플래그를 정의하고, 이러한 플래그는 작은 폭의 잔차값을 충분히 전송한다. 예컨대, sig_coeff_flag는 잔차 폭이 0인지 여부를 지시한다. sig_coeff_flag가 1이면, 더욱 많은 플래그(일반적으로 abs_level_gtx_flag로 명명함)를 전송할 수 있고, 이러한 플래그는 잔차 폭이 1, 2, 3 등보다 큰지 여부를 지시한다. 이러한 플래그는 CABAC 엔진에 의해 콘텐츠 코딩된다. 대부분 잔차 계수의 폭이 비교적 작으므로, CABAC는 상대적으로 비교적 적은 빈을 사용만 하면 대부분 잔차에 대해 유효 코딩을 수행할 수 있다.

[0187] 잔차 계수 플래그를 통해 전송될 수 없는 잔차 계수의 임의의 나머지 폭은 모두 신택스 요소 abs_remainder에서 전송된다. abs_remainder의 값이 $6*2^R$보다 작거나 같으면, 상기 신택스 요소는 전부 abs_remainder의 라이스 코딩을 통해 전송된다. $6*2^R$보다 크면, 상기 신택스 요소는 $6*2^R$의 라이스 코드 및 $(\text{abs_remainder}-6*2^R)$의 지수 골롬 코딩의 결합(concatenation)을 통해 전송된다. 여기서 지수 골롬 코딩 과정에 대해 설명하지 않는다.

[0188] 라이스 코딩이 CABAC에 비해 간단하지만, 적합한 조건 하에서, 라이스 코딩도 저비트레이트로 유효하게 압축될 수 있다. 값이 전부 비교적 작은 잔차 계수의 경우, 비교적 작은 라이스 파라미터를 구비하는 라이스 코딩 효율

이 더욱 높다. 반면, 특정된 잔차 계수의 값이 비교적 클 경우, 비교적 큰 라이스 파라미터가 비교적 적합할 수 있다. 잔차 계수의 데이터 통계를 위해, 값 locSumAbs을 통해 적응적으로 라이스 파라미터를 결정할 수 있고, 상기 값은 인접 잔차 계수의 폭에 따라 계산하여 획득된 것이다.

locSumAbs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cRiceParam	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2
locSumAbs	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cRiceParam	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3

[0189]

[0190] 적응 라이스 파라미터 결정은 “정규 잔차 코딩”(RRC)에서의 잔차 계수를 위해 설계된 것이고, RRC에서의 잔차 계수는 이산 코사인 변환(DCT)을 실행하는 것을 통해 획득된 잔차 계수이다. 그러나, VVCv1에서, 적응 라이스 파라미터 결정은 “변환 스킵 잔차 코딩”(TSRC)에서의 잔차 계수에도 적용된다. VVCv2에서, 라이스 파라미터에 의해 결정된 대안적인 메커니즘이 변환-스킵 계수에 유익할 수 있는 것을 인식하였다.

[0191] VVCv2에서의 대안적인 메커니즘은 슬라이스 레벨 신택스 요소 sh_ts_residual_coding_rise_idx_minus1를 통해 라이스 파라미터를 명확하게 전송하는 것을 허용한다. 상기 신택스 요소를 전송할 경우, 라이스 파라미터는 R=sh_ts_residual_coding_rise_idx_minus1+1로 설정된다. 라이스 파라미터의 이러한 값은 슬라이스 지속 시간 내에서 계속 유지된다.

if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
if(!sh_ts_residual_coding_disabled_flag && sps_ts_residual_coding_rice_present_in_sh_flag)	
sh_ts_residual_coding_rice_idx_minus1	u(3)
if(sps_reverse_last_sig_coeff_enabled_flag)	
sh_reverse_last_sig_coeff_flag	u(1)
if(pps_slice_header_extension_present_flag) {	
sh_slice_header_extension_length	ue(v)

[0192]

[0193] Sh_ts_residual_coding_rice_idx_minus1 더하기 1은 현재 슬라이스 중의 residual_ts_coding() 신택스 구조에 사용되는 라이스 파라미터를 규정한다. 존재하지 않을 경우, sh_ts_residual_coding_rice_idx_minus1의 값은 0과 같은 것으로 추론된다.

[0194] 대안적인 메커니즘을 인에이블할지 여부는 SPS 레벨 플래그 sps_ts_residual_coding_rice_present_in_sh_flag에 의해 제어된다. 상기 플래그가 0으로 설정되면, 대안적인 라이스 파라미터 전송을 인에이블하지 않는다. GCI 플래그 gci_no_ts_residual_coding_rice_constraint_flag는 하이 레벨에서 명시적 라이스 파라미터 전송의 VVCv2 툴이 제약되는지 여부를 전송한다. gci_no_ts_residual_coding_rice_constraint_flag가 1과 같으면 OlsInScope 중 모든 픽처의 sps_ts_residual_coding_rice_present_in_sh_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_ts_residual_coding_rice_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_ts_residual_coding_rice_constraint_flag가 존재하지 않을 경우,

gci_no_ts_residual_coding_rice_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0195] **gci_no_rrc_rice_extension_constraint_flag**

[0196] gci_no_rrc_rice_extension_constraint_flag는 OlsInScope 중 모든 픽처의 sps_rrc_rice_extension_flag를 규정한다. gci_no_rrc_rice_extension_constraint_flag가 1과 같으면 OlsInScope 중 모든 픽처의 sps_rrc_rice_extension_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_rrc_rice_extension_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다.

[0197] 고 비트 깊이 및 고 비트 레이트의 애플리케이션의 경우, RRC의 많은 위치에 모두 많은 큰 양자화 레벨이 존재한다. 이러한 타입의 애플리케이션의 경우, 비교적 큰 라이스 파라미터는 나머지 레벨을 나타내는데 필요한 빈 개수가 적어지는 것을 초래한다. VVCv1 중 라이스 파라미터를 유도하는 방법은 VVCv2 애플리케이션에 있어서 최적인 것이 아닐 수 있다. 따라서, 대안적인 라이스 파라미터 유도를 사용할 수 있으며, 대안적인 라이스 파라미터 유도는 sps_rrc_rice_extension_flag에 의해 전송될 수 있다. sps_rrc_rice_extension_flag가 1과 같으면 abs_remaining[ ] 및 dec_abs_level[ ]의 이진화에 대해 대안적인 라이스 파라미터 유도를 사용하는 것으로 규정한다. sps_rrc_rice_extension_flag가 0과 같으면 abs_remaining[ ] 및 dec_abs_level[ ]의 이진화에 대해 대안적인 라이스 파라미터 유도를 사용하지 않는 것으로 규정한다. 존재하지 않을 경우, sps_rrc_rice_extension_flag의 값은 0과 같은 것으로 추론된다.

[0198] 아래에서 VVCv2에서 sps_rrc_rice_extension_flag를 사용하여 라이스 파라미터를 결정하는 예를 도시한다. 성분 인덱스 cIdx를 구비하는 변환 블록의 어레이 AbsLevel[x][y] 및 왼쪽 위 휘도 위치(x0, y0)가 주어지고, 아래의 의사 코드 과정의 규정에 따라 변수 locSumAbs를 유도한다(밑줄 그은 부분은 VVCv1에 비해 VVCv2가 증가하는 콘텐츠임).

```
locSumAbs = 0

if( xC < ( 1 << AbsLog2TbWidth ) - 1 ) {
    locSumAbs += AbsLevel[ xC + 1 ][ yC ]
    if( xC < ( 1 << AbsLog2TbWidth ) - 2 )
        locSumAbs += AbsLevel[ xC + 2 ][ yC ]
    else
        locSumAbs += HistValue
    if( yC < ( 1 << AbsLog2TbHeight ) - 1 )
        locSumAbs += AbsLevel[ xC + 1 ][ yC + 1 ]        (1517)
    else
        locSumAbs += HistValue
} else
    locSumAbs += 2 * HistValue
    if( yC < ( 1 << AbsLog2TbHeight ) - 1 ) {
    locSumAbs += AbsLevel[ xC ][ yC + 1 ]
    if( yC < ( 1 << AbsLog2TbHeight ) - 2 )
        locSumAbs += AbsLevel[ xC ][ yC + 2 ]
```

[0199]

```
    else
        locSumAbs += HistValue
} else
    locSumAbs += HistValue
```

리스트 Tx[ ] 및 Rx[ ] 규정은 아래와 같다.

Tx[ ] = { 32, 128, 512, 2048 } (1523)

Rx[ ] = { 0, 2, 4, 6, 8 } (1524)

변수 shiftVal의 값 유도는 아래와 같다.

```
if( !sps_rrc_rice_extension_flag )
    shiftVal = 0                                    (1526X1)
else
 shiftVal = ( localSumAbs < Tx[ 0 ] ) ? Rx[ 0 ] : ( ( localSumAbs
< Tx[ 1 ] ) ? Rx[ 1 ] :
    ( ( localSumAbs < Tx[ 2 ] ) ? Rx[ 2 ] : ( ( localSumAbs <
Tx[ 3 ] ) ? Rx[ 3 ] : Rx[4] ) ) )
```

locSumAbs 의 값 업데이트는 아래와 같다.

[0200]

[0201] locSumAbs = Clip3(0, 31, (locSumAbs >> shiftVal) - baseLevel * 5) (1526X2)

[0202] 변수 locSumAbs가 주어지고, 먼저 표 128에서 규정된 것에 따라 라이스 파라미터 cRiceParam를 유도한 다음, 아래와 같이 업데이트 한다.

[0203] cRiceParam = cRiceParam + shiftVal (1526X3)

[0204] baseLevel이 0과 같을 경우, 변수 ZeroPos[n]의 유도는 아래와 같다.

[0205] ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam (1518)

[0206] **표 128-locSumAbs에 기반한 cRiceParam 규범**

locSumAbs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cRiceParam	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2
locSumAbs	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
cRiceParam	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3

[0207]

[0208] 식 (1526X3)으로부터 볼 수 있다 시피, VVCv1에 비해, VVCv2에서 절대 레벨의 나머지 부분을 이진화하기 위한 cRiceParam이 더욱 클 수 있다.

[0209] **gci_no_persistent_rice_adaptation_constraint_flag**

[0210] gci_no_persistent_rice_adaptation_constraint_flag는 라이스 파라미터 유도에 대한 제약을 전송하고, 상기 라이스 파라미터 유도는 이전 TU 상태를 사용하는 이진화에 대한 것이다.

gci_no_persistent_rice_adaptation_constraint_flag가 1과 같으면 OlsInScope 중 모든 픽처의 sps_persistent_rice_adaptation_enabled_flag가 모두 0과 같아야 하는 것으로 규정한다. gci_no_persistent_rice_adaptation_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_persistent_rice_adaptation_constraint_flag가 존재하지 않을 경우, gci_no_persistent_rice_adaptation_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0211] sps_persistent_rice_adaptation_enabled_flag가 1과 같으면 각 TU가 시작될 경우, 이전 TU에서 누적된 데이터 통계를 사용하여 abs_remainder[ ] 및 dec_abs_level[ ]를 이진화하기 위한 라이스 파라미터 유도를 초기화하는 것으로 규정한다. sps_persistent_rice_adaptation_enabled_flag가 0과 같으면 라이스 파라미터 유도에서 이전의 TU 상태를 사용하지 않는 것으로 규정한다. 존재하지 않을 경우, sps_persistent_rice_adaptation_enabled_flag의 값은 0과 같은 것으로 추론된다. 아래에서, VVCv2에서 sps_persistent_rice_adaptation_enabled_flag를 사용하여 라이스 파라미터를 결정하는 예를 도시한다(밑줄 그은 부분은 VVCv1에 비해 VVCv2가 증가하는 콘텐츠임).

[0212] CTU가 슬라이스 또는 타일 tile에서의 첫 번째 CTU이면, 서브 조항 9.3.2.2에서 규정된 콘텐츠 변수를 호출하는 초기화 과정에 따라, 어레이 PredictorPaletteSize[chType](chType = 0, 1)를 0으로 초기화하고, 어레이 StatCoeff[i](i = 0...2)를 아래와 같이 초기화한다.

[0213] StatCoeff[i] = sps_persistent_rice_adaptation_enabled_flag ? 2 * Floor(Log2(BitDepth - 10)) : 0 (1513X3)

[0214] StatCoeff[i]는 HisValue를 계산하는데 사용되고, HisValue는 식 (1517)에서의 locSumAbs를 계산하는데 사용되며, HisValue는 각 TU에서 한회 업데이트될 수 있다. HisValue에 의해, 블록 경계 위치에 위치하는 라이스 파라미터의 유도값으로 하여금 더욱 정확하도록 할 수 있다.

[0215] **gci_no_reverse_last_sig_coeff_constraint_flag**

[0216] gci_no_reverse_last_sig_coeff_constraint_flag가 1과 같으면 OlsInScope 중 모든 픽처의 sps_reverse_last_sig_coeff_enabled_flag가 모두 0과 같아야 하는 것으로 규정된다. gci_no_reverse_last_sig_coeff_constraint_flag가 0과 같으면 이러한 제약을 가하지 않는다. gci_no_reverse_last_sig_coeff_constraint_flag가 존재하지 않을 경우, gci_no_reverse_last_sig_coeff_constraint_flag의 값은 0과 같은 것으로 추론된다.

[0217] 정규 잔차 코딩(RRC)에서, TU 중 마지막 0이 아닌 레벨의 위치(x, y)는 최대 4 개의 신택스 요소, 즉 last_sig_coeffx_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix 및 last_sig_coeff_y_suffix에 의해 코딩된다. VVCv1 중 현재 TU의 (x, y)와 (0, 0) 사이의 차이값을 사용하여 상기 위치를 코딩한다. 이것은 VVCv1에 대해 합리한 것이고, 그 원인은 하나의 TU 내에 많은 0 레벨이 존재하고, 대다수 0이 아닌 레벨이 모두 TU의 왼쪽 위 코너에 위치하기 때문이다. 그러나, VVCv2 애플리케이션의 경우, 그렇지 않을 수 있으며, 많은 0이 아닌 레벨이 전체 TU에 분포되므로, 왼쪽 위 코너 (0, 0)이 아닌 오른 쪽 아래 코너에 대한 상기 위치 (x, y)를 코딩하는 것이 유익할 수 있다. sh_reverse_last_sig_coeff_flag는 이러한 타입의 애플리케이션을 처리하는 툴을 제공한다.

[0218] sh_reverse_last_sig_coeff_flag가 1과 같으면 현재 슬라이스의 각 변환 블록에 대해, ((Log2ZoTbWidth << 1) - 1, (Log2ZoTbHeight << 1) - 1)에 대한 마지막 유효 계수(significant coefficient)의 좌표를 코딩하는 것으로 규정한다. sh_reverse_last_sig_coeff_flag가 0과 같으면 현재 슬라이스의 각 변환 블록에 대해, (0, 0)에 대한 마지막 유효 계수의 좌표를 코딩한다. 존재하지 않을 경우, sh_reverse_last_sig_coeff_flag의 값은 0과 같은 것으로 추론된다.

[0219] 아래와 같이, 슬라이스 헤드에서, sps_reverse_last_sig_coeff_enabled_flag를 이용하여 sh_reverse_last_sig_coeff_flag에 대해 조건부 파싱을 수행한다.

[0220] -last_sig_coeff_x_suffix가 존재하지 않으면, 아래의 공식에 적용된다.

[0221] LastSsignificantCoeffX = last_sig_coeff_x_prefix (193)

[0222] -그렇지 않은 경우 (last_sig_coeff_x_suffix가 존재함), 아래의 공식에 적용된다.

[0223] LastSignificantCoeffX = (1 << ((last_sig_coeff_x_prefix >> 1) - 1)) * (194)

[0224] (2 + (last_sig_coeff_x_prefix & 1)) + last_sig_coeff_x_suffix

[0225] sh_reverse_last_sig_coeff_flag가 1과 같을 경우, LastSignificantCoeffX의 값 수정은 아래와 같다.

[0226] LastSignificantCoeffX = (1 << Log2ZoTbWidth) - 1 - LastSignificantCoeffX (195)

[0227] -last_sig_coeff_y_suffix가 존재하지 않으면, 아래의 공식에 적용된다.

[0228] LastSignificantCoeffY = last_sig_coeff_y_prefix (196)

[0229] -그렇지 않은 경우(last_sig_coeff_y_suffix가 존재함), 아래의 공식에 적용된다.

[0230] LastSignificantCoeffY = (1 << ((last_sig_coeff_y_prefix >> 1) - 1)) * (197)

[0231] (2 + (last_sig_coeff_y_prefix & 1)) + last_sig_coeff_y_suffix

[0232] sh_reverse_last_sig_coeff_flag가 1과 같을 경우, LastSignificantCoeffY의 값 수정은 아래와 같다.

[0233] LastSignificantCoeffY = (1 << Log2ZoTbHeight) - 1 - LastSignificantCoeffY (198)

[0234] VVCv2 슬라이스 헤드

if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
if(!sh_ts_residual_coding_disabled_flag && sps_ts_residual_coding_rice_present_in_sh_flag)	
sh_ts_residual_coding_rice_idx_minus1	u(3)
if(sps_reverse_last_sig_coeff_enabled_flag)	
sh_reverse_last_sig_coeff_flag	u(1)
if(pps_slice_header_extension_present_flag) {	
sh_slice_header_extension_length	ue(v)

[0235]

[0236] 도 5는 본 발명의 일부 실시예를 따른 비디오 디코딩의 과정(500)의 예를 설명한다. 하나 또는 복수 개의 컴퓨팅 기기는 적합한 프로그램 코드를 실행하는 것을 통해 도 5에서 설명된 동작을 구현한다. 예컨대, 비디오 디코더(200)를 구현하는 컴퓨팅 기기는 프로그램 코드를 실행하는 것을 통해 도 5에서 설명된 동작을 구현할 수 있다. 상기 컴퓨팅 기기는, 예컨대, 엔트로피 디코딩 모듈(216), 역양자화 모듈(218), 역변환 모듈(219), 인루프 필터 모듈(220), 인터 예측 모듈(224) 및 인트라 예측 모듈(226)을 포함한다. 설명하기 위해, 도면에서 설명된 일부 예를 참조하여 과정(500)을 설명한다. 그러나, 다른 구현 방식을 사용할 수도 있다.

[0237] 단계 502에서, 과정(500)은 코딩 비디오(202)와 같은 비디오 신호의 비트 스트림을 액세스하는데 관한 것이다. 단계504에서, 과정(500)은 비디오의 비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출하는데 관한 것이다. 전술한 바와 같이, 이진 플래그 gci_present_flag는 GCI 신택스 요소가 존재하는지 여부를 규정하기 위한 것이다. gci_present_flag가 1과 같으면 general_constraints_info() 신택스 구조에 GCI 신택스 요소가 존재하는 것으로 규정하고, GCI 신택스 요소는 부가 코딩 툴에 가한 제약을 지시하기 위한 것이다. gci_present_flag가 0과 같으면 GCI 신택스 요소가 존재하지 않고, 비디오에 일반 제약을 가하지 않는 것으로 규정한다. 인코더의 상이함에 따라, 비디오의 네트워크 데이터 패킷, 비디오의 비디오 파라미터 세트 또는 비디오의 시퀀스 파라

미터 세트로부터 GCI 플래그를 추출할 수 있다.

[0238] 단계506에서, 과정(500)은 GCI 플래그의 값에 따라 비디오에 하나 또는 복수 개의 일반 제약을 가하였는지 여부를 결정하는데 관한 것이다. 비디오에 하나 또는 복수 개의 일반 제약을 가하였으면(즉, GCI 플래그가 1임), 단계 508에서, 과정(500)은 비디오의 비트 스트림으로부터 비디오 비트 스트림에 포함된 부가 비트 개수를 나타내는 값 M을 추출하는데 관한 것이다. 이러한 부가 비트는 플래그 비트를 포함하며, 각 부가 코딩 툴이 상기 비디오의 경우 제약되는 것을 각각 지시한다.

[0239] 단계 510에서, 과정(500)은 M이 5보다 큰지 여부를 결정하는데 관한 것이다. M이 5보다 크면, 단계 512에서, 과정(500)은 비트 스트림으로부터 6 개의 플래그 비트를 추출하고, 상기 6 개의 플래그 비트는 6 개의 부가 코딩 툴의 각각의 제약을 지시하는 플래그를 각각 나타낸다. 상기 6 개의 플래그는, 비디오의 픽처가 인트라 랜덤 액세스 포인트(IRAP) 픽처 또는 점진적 디코딩 리프레시(GDR) 픽처로 한정되는 것을 지시하는 플래그 **gci_all_rap_pictures_constraint_flag**; 확장 변환 정밀도를 제약하는지 여부를 지시하는 플래그 **gci_no_extended_precision_processing_constraint_flag**; 명시적 라이스 파라미터 전송을 제약하는지 여부를 지시하는 플래그 **gci_no_ts_residual_coding_rice_constraint_flag**; 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도를 지시하는 플래그 **gci_no_rrc_rice_extension_constraint_flag**; 이전의 변환 유닛에 기반한 이진화를 위한 라이스 파라미터 유도를 초기화하는지 여부를 지시하는 플래그 **gci_no_persistent_rice_adaptation_constraint_flag**; 및 TU 중 마지막 0이 아닌 레벨의 위치를 디코딩할 경우, OlsInScope에서의 픽처에 제약을 가하는지 여부를 지시하는 플래그 **gci_no_reverse_last_sig_coeff_constraint_flag**를 포함한다.

[0240] M이 6보다 크면, 단계 513에서, 과정(500)은 비트 스트림으로부터 나머지의 M-6 개의 비트를 추출하고 버리는데 관한 것이다. 다시 말해서, 비디오의 디코딩은 상기 M-6 개의 비트와 독립적으로 수행된다.

[0241] 단계 514에서, 과정(500)은6 개의 부가 코딩 툴을 위해 6 개의 플래그가 지시한 제약에 따라, 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는데 관한 것이다. 예컨대, 플래그 **gci_all_rap_pictures_constraint_flag**가 1이면, 디코더는 하나 또는 복수 개의 출력층 세트에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 결정하고, 하나 또는 복수 개의 출력층 세트에서의 GDR 픽처 또는 IRAP 픽처를 디코딩할 수 있다. 플래그 **gci_no_extended_precision_processing_constraint_flag**가 1이면, 디코더는 확장 변환 정밀도가 제약되고, OlsInScope에서의 픽처의 sps_extended_precision_flag를 0과 같도록 설정하여 확장 동적 범위를 사용하지 않는 것을 통해, 비디오를 디코딩할 수 있다. 플래그 **gci_no_ts_residual_coding_rice_constraint_flag**가 1이면, 디코더는 명시적 라이스 파라미터 전송이 제약되는 것으로 결정하고, OlsInScope에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다. 플래그 **gci_no_rrc_rice_extension_constraint_flag**가 1이면, 디코더는 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도가 제약되는 것으로 결정하고, OlsInScope에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다. 플래그 **gci_no_persistent_rice_adaptation_constraint_flag**가 1이면, 디코더는 이전의 변환 유닛 상태에 기반한 이진화를 위한 라이스 파라미터 유도에 대한 초기화가 제약되는 것으로 결정하고, OlsInScope에서의 픽처의 경우 이전의 변환 유닛 상태에 따라 라이스 파라미터를 초기화하지 않는 경우, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다. 플래그 **gci_no_reverse_last_sig_coeff_constraint_flag**가 1이면, 디코더는 OlsInScope 중 모든 픽처의 sps_reverse_last_sig_coeff_enabled_flag가 모두 0이고, 현재 슬라이스의 각 변환 블록의 경우, 마지막 유효 계수의 좌표는 모두 왼쪽 위 코너(0, 0)에 대해 코딩된 것으로 결정하며; 마지막 유효 계수의 디코딩된 좌표를 현재 슬라이스의 각 변환 블록에 대한 왼쪽 위 코너(0, 0)로 해석하는 것을 통해, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다.

[0242] 단계 510에서 M이 5보다 크지 않는 것으로 결정하면, 단계 518에서, 과정(500)은 비트 스트림으로부터 M 개의 비트를 추출하고 버리는데 관한 것이다. 다시 말해서, 비디오의 디코딩은 상기 M 개의 비트와 독립적으로 수행된다. 단계 520에서, 디코더는 6 개의 부가 코딩 툴에 제약을 가하지 않는 경우 비디오를 디코딩한다. 단계 506에서 GCI 플래그가 비디오에 일반 제약을 가하지 않는 것을 지시하는 것으로 결정하면(즉, GCI 플래그가 0임), 과정(500)은 일반 제약이 없는 경우 비디오를 이미지로 디코딩하는데 관한 것이다. 일부 예에서, 도 2에 대한 상기 과정을 통해 디코딩을 실행할 수 있다. 디코딩 비디오를 출력하여 디스플레이하는데 사용할 수 있다.

[0243] 도 6은 본 발명의 일부 실시예에 따른 비디오 디코딩의 과정(600)의 다른 하나의 예를 설명한다. 하나 또는 복

수 개의 컴퓨팅 기기는 적합한 프로그램 코드를 실행하는 것을 통해 도 6에서 설명된 동작을 구현한다. 예컨대, 비디오 디코더(200)를 구현하는 컴퓨팅 기기는 프로그램 코드를 실행하는 것을 통해 도 6에서 설명된 동작을 구현할 수 있다. 상기 컴퓨팅 기기는, 예컨대, 엔트로피 디코딩 모듈(216), 역양자화 모듈(218), 역변환 모듈(219), 인루프 필터 모듈(220), 인터 예측 모듈(224) 및 인트라 예측 모듈(226)을 포함한다. 설명하기 위해, 도면에서 설명된 일부 예를 참조하여 과정(600)을 설명한다. 그러나, 다른 구현 방식을 사용할 수도 있다.

[0244] 단계 602에서, 과정(600)은 코딩 비디오(202)와 같은 비디오 신호의 비트 스트림을 액세스하는데 관한 것이다. 단계604에서, 과정(600)은 비디오의 비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출하는데 관한 것이다. 전술한 바와 같이, 이진 플래그 gci_present_flag는 GCI 신택스 요소가 존재하는지 여부를 규정하기 위한 것이다. gci_present_flag가 1과 같으면 general_constraints_info() 신택스 구조에 GCI 신택스 요소가 존재하는 것으로 규정하고, GCI 신택스 요소는 부가 코딩 툴에 가한 제약을 지시하기 위한 것이다. gci_present_flag가 0과 같으면 GCI 신택스 요소가 존재하지 않는 않고, 비디오에 일반 제약을 가하지 않는 것으로 규정한다. 인코더의 상이함에 따라, 비디오의 네트워크 데이터 패킷, 비디오의 비디오 파라미터 세트 또는 비디오의 시퀀스 파라미터 세트로부터 GCI 플래그를 추출할 수 있다.

[0245] 단계606에서, 과정(600)은 GCI 플래그의 값에 따라 비디오에 하나 또는 복수 개의 일반 제약을 가하였는지 여부를 결정하는데 관한 것이다. 비디오에 하나 또는 복수 개의 일반 제약을 가하였으면(즉, GCI 플래그가 1임), 단계 608에서, 과정(600)은 비디오의 비트 스트림으로부터 비디오 비트 스트림에 포함된 부가 비트 개수를 나타내는 값 M을 추출하는데 관한 것이다. 이러한 부가 비트는 플래그 비트를 포함하며, 각 부가 코딩 툴이 상기 비디오의 경우 제약되는 것을 각각 지시한다.

[0246] 단계 610에서, 과정(600)은 M이 6보다 큰지 여부를 결정하는데 관한 것이다. M이 6보다 크면, 단계 612에서, 과정(600)은 비트 스트림으로부터 6 개의 플래그 비트를 추출하고, 상기 6 개의 플래그 비트는 6 개의 부가 코딩 툴의 각각의 제약을 지시하는 플래그를 각각 나타낸다. 상기 6 개의 플래그는, 비디오의 픽처가 인트라 랜덤 액세스 포인트(IRAP) 픽처 또는 점진적 디코딩 리프레시(GDR) 픽처로 한정되는 것을 지시하는 플래그 **gci_all_rap_pictures_constraint_flag**; 확장 변환 정밀도를 제약하는지 여부를 지시하는 플래그 **gci_no_extended_precision_processing_constraint_flag**; 명시적 라이스 파라미터 전송을 제약하는지 여부를 지시하는 플래그 **gci_no_ts_residual_coding_rice_constraint_flag**; 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도를 지시하는 플래그 **gci_no_rrc_rice_extension_constraint_flag**; 이전의 변환 유닛에 기반한 이진화를 위한 라이스 파라미터 유도를 초기화하는지 여부를 지시하는 플래그 **gci_no_persistent_rice_adaptation_constraint_flag**; 및 TU 중 마지막 0이 아닌 레벨의 위치를 디코딩할 경우, OlsInScope에서의 픽처에 제약을 가하는지 여부를 지시하는 플래그 **gci_no_reverse_last_sig_coeff_constraint_flag**를 포함한다.

[0247] 단계 614에서, 과정(600)은 비트 스트림으로부터 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출하고, 추출된 M-6 개의 비트를 버리는데 관한 것이다. 다시 말해서, 비디오의 디코딩은 상기 M-6 개의 비트와 독립적으로 수행된다. 단계 616에서, 과정(600)은 6 개의 부가 코딩 툴을 위해6 개의 플래그가 지시한 제약에 따라, 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는데 관한 것이다. 예컨대, 플래그 **gci_all_rap_pictures_constraint_flag**가 1이면, 디코더는 하나 또는 복수 개의 출력층 세트에서의 모든 픽처가 모두 ph_recovery_poc_cnt가 0인 GDR 픽처, 또는 IRAP 픽처인 것으로 결정하고, 하나 또는 복수 개의 출력층 세트에서의 GDR 픽처 또는 IRAP 픽처를 디코딩할 수 있다. 플래그 **gci_no_extended_precision_processing_constraint_flag**가 1이면, 디코더는 확장 변환 정밀도가 제약되고, OlsInScope에서의 픽처의 sps_extended_precision_flag를 0과 같도록 설정하여 확장 동적 범위를 사용하지 않는 것을 통해, 비디오를 디코딩할 수 있다. 플래그 **gci_no_ts_residual_coding_rice_constraint_flag**가 1이면, 디코더는 명시적 라이스 파라미터 전송이 제약되는 것으로 결정하고, OlsInScope에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다. 플래그 **gci_no_rrc_rice_extension_constraint_flag**가 1이면, 디코더는 비디오의 양자화 잔차 이진화를 위한 대안적인 라이스 파라미터 유도가 제약되는 것으로 결정하고, OlsInScope에서의 픽처의 대안적인 라이스 파라미터 전송을 사용 금지하는 것을 통해, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다. 플래그 **gci_no_persistent_rice_adaptation_constraint_flag**가 1이면, 디코더는 이전의 변환 유닛 상태에 기반한 이진화를 위한 라이스 파라미터 유도에 대한 초기화가 제약되는 것으로 결정하고, OlsInScope에서의 픽처의 경우 이전의 변환 유닛 상태에 따라 라이스 파라미터를 초기화하지 않는 경우, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다. 플래그 **gci_no_reverse_last_sig_coeff_constraint_flag**가 1이면, 디코더는 OlsInScope 중

모든 픽처의 sps_reverse_last_sig_coeff_enabled_flag가 모두 0이고, 현재 슬라이스의 각 변환 블록의 경우, 마지막 유효 계수의 좌표는 모두 왼쪽 위 코너(0, 0)에 대해 코딩된 것으로 결정하며; 마지막 유효 계수의 디코딩된 좌표를 현재 슬라이스의 각 변환 블록에 대한 왼쪽 위 코너(0, 0)로 해석하는 것을 통해, 비디오의 비트 스트림의 나머지 부분을 디코딩할 수 있다.

[0248] 단계 610에서 M이 6보다 크지 않은 것으로 결정하면, M은 0 또는 6이다. M이 6이면, 단계 620에서, 과정(600)은 상기 6 개의 플래그 비트를 추출하는데 관한 것이다. 단계 622에서, 디코더는 6 개의 부가 코딩 툴에 제약을 가하는 것을 통해, 추출된 플래그 비트에 따라 비디오를 디코딩한다. M이 0이면, 플래그 비트를 추출하지 않는다. 단계 624에서, 플래그 비트를 추출하지 않았기 때문에, 디코더는 6 개의 부가 코딩 툴에 제약을 가하지 않는 것을 통해 비디오를 디코딩한다. 단계 606에서 GCI 플래그가 비디오에 일반 제약을 가하지 않는 것을 지시하는 것으로 결정하면(즉, GCI 플래그가 0임), 단계 618에서, 과정(600)은 일반 제약이 없는 경우 비디오를 이미지로 디코딩하는데 관한 것이다. 일부 예에서, 도 2에 대한 상기 과정을 통해 디코딩을 실행할 수 있다. 디코딩 비디오를 출력하여 디스플레이하는데 사용할 수 있다.

[0249] 도 7은 본 발명의 일부 실시예에 따른 비디오 디코딩의 과정(700)의 다른 하나의 예를 설명한다. 하나 또는 복수 개의 컴퓨팅 기기는 적합한 프로그램 코드를 실행하는 것을 통해 도 7에서 설명된 동작을 구현한다. 예컨대, 비디오 디코더(200)를 구현하는 컴퓨팅 기기는 프로그램 코드를 실행하는 것을 통해 도 7에서 설명된 동작을 구현할 수 있다. 상기 컴퓨팅 기기는, 예컨대, 엔트로피 디코딩 모듈(216), 역양자화 모듈(218), 역변환 모듈(219), 인루프 필터 모듈(220), 인터 예측 모듈(224) 및 인트라 예측 모듈(226)을 포함한다. 설명하기 위해, 도면에서 설명된 일부 예를 참조하여 과정(700)을 설명한다. 그러나, 다른 구현 방식을 사용할 수도 있다.

[0250] 단계 702에서, 과정(700)은 코딩 비디오(202)와 같은 비디오 신호의 비트 스트림을 액세스하는데 관한 것이다. 단계704에서, 과정(700)은 비디오의 비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출하는데 관한 것이다. 전술한 바와 같이, 이진 플래그 gci_present_flag는 GCI 신택스 요소가 존재하는지 여부를 규정하기 위한 것이다. gci_present_flag가 1과 같으면 general_constraints_info() 신택스 구조에 GCI 신택스 요소가 존재하는 것으로 규정하고, GCI 신택스 요소는 부가 코딩 툴에 가한 제약을 지시하기 위한 것이다. gci_present_flag가 0과 같으면 GCI 신택스 요소가 존재하지 않는 않고, 비디오에 일반 제약을 가하지 않는 것으로 규정한다. 인코더의 상이함에 따라, 비디오의 네트워크 데이터 패킷, 비디오의 비디오 파라미터 세트 또는 비디오의 시퀀스 파라미터 세트로부터 GCI 플래그를 추출할 수 있다.

[0251] 단계 706에서, 과정(700)은 GCI 플래그의 값에 따라 비디오에 하나 또는 복수 개의 일반 제약을 가하였는지 여부를 결정하는데 관한 것이다. 비디오에 하나 또는 복수 개의 일반 제약을 가하였으면(즉, GCI 플래그가 1임), 단계 708에서, 과정(700)은 비디오의 비트 스트림으로부터 비디오 비트 스트림에 포함된 부가 비트 개수를 나타내는 값 M을 추출하는데 관한 것이다. 이러한 부가 비트는 플래그 비트를 포함하며, 각 부가 코딩 툴이 상기 비디오의 경우 제약되는 것을 각각 지시한다.

[0252] 단계 710에서, 과정(700)은 M이 5보다 큰지 여부를 결정하는데 관한 것이다. M이 5보다 크지 않으면, 단계 712에서, 과정(700)은 비트 스트림으로부터 M 개의 비트를 추출하고 상기 M 개의 비트를 버리는데 관한 것이다. 단계 714에서, 과정(700)은 상기 M 개의 비트와 독립적으로, 비디오의 비트 스트림의 나머지 부분을 이미지로 디코딩하는데 관한 것이다. 단계 710에서 M이 5보다 크지 않은 것으로 결정하면, 단계 718에서, 과정(700)은 상기 6 개의 플래그 비트를 추출하는데 관한 것이다. M이 6보다 크면, 단계 719에서, 과정(700)은 비트 스트림으로부터 나머지의 M-6 개의 비트를 추출하고 버리는데 관한 것이다. 다시 말해서, 비디오의 디코딩은 상기 M-6 개의 비트와 독립적으로 수행된다. 단계 720에서, 전술한 바와 같이, 디코더는 6 개의 부가 코딩 툴에 제약을 가하는 것을 통해, 추출된 6 개의 플래그 비트에 따라 비디오를 디코딩한다.

[0253] 단계 706에서 GCI 플래그가 비디오에 일반 제약을 가하지 않는 것을 지시하는 것으로 결정하면(즉, GCI 플래그가 0임), 과정(700)은 일반 제약이 없는 경우 비디오를 이미지로 디코딩하는데 관한 것이다. 일부 예에서, 도 2에 대한 상기 과정을 통해 디코딩을 실행할 수 있다. 디코딩 비디오를 출력하여 디스플레이하는데 사용할 수 있다.

[0254] *일반 제약 정보 전송을 구현하는 컴퓨팅 시스템의 예*

[0255] 임의의 적합한 컴퓨팅 시스템은 모두 본 문에서 전술한 동작을 실행하는데 사용될 수 있다. 예컨대, 도 8은 도 1의 비디오 인코더(100) 또는 도 2의 비디오 디코더(200)를 구현할 수 있는 컴퓨팅 기기(800)의 예를 설명한다. 일부 실시예에서, 컴퓨팅 기기(800)는 프로세서(812)를 포함할 수 있고, 프로세서(812)는 메모리(814)에 통신적

으로 커플링되고, 컴퓨터 실행 가능한 프로그램 코드를 실행하는 것 및 메모리(814)에 저장된 정보를 액세스하는 것 중 적어도 하나를 수행한다. 프로세서(812)는 마이크로 프로세서, 전용 집적 회로(“ASIC”), 상태 머신 또는 다른 처리 기기를 포함할 수 있다. 프로세서(812)는 다양한 처리 기기에서의 어느 하나 또는 복수 개를 포함할 수 있다. 이러한 프로세서는 명령어를 저장하는 컴퓨터 판독 가능한 매체를 포함할 수 있고, 또는 명령어를 저장하는 컴퓨터 판독 가능한 매체와 통신할 수 있으며, 이러한 명령어는 프로세서에 의해 실행될 경우 프로세서로 하여금 본 문에서 설명된 단계를 실행하도록 한다.

[0256] 메모리(814)는 임의의 적합한 비 일시적 컴퓨터 판독 가능한 매체를 포함할 수 있다. 컴퓨터 판독 가능한 매체는 프로세서에 컴퓨터 판독 가능한 명령어 또는 다른 프로그램 코드를 제공할 수 있는 임의의 전자, 광학, 자기 또는 다른 저장 기기를 포함할 수 있다. 컴퓨터 판독 가능한 매체의 비 제한적인 예는 자기 디스크, 메모리 칩, ROM, RAM, ASIC, 구성된 프로세서, 광학 스토리지, 자기 테이프 또는 다른 자기 스토리지, 또는 컴퓨터프 로세서가 명령어를 판독할 수 있는 임의의 다른 매체를 포함한다. 이러한 명령어는 컴파일러 및 인터프리터 중 적어도 하나에 의해 임의의 적합한 컴퓨터 프로그래밍 언어(예컨대, C, C++, C#, Visual Basic, Java, Python, Perl, JavaScript 및 ActionScript를 포함함)로 작성된 코드에 의해 생성된 프로세서 전용 명령어를 포함할 수 있다.

[0257] 컴퓨팅 기기(800)는 또한 버스(816)를 포함할 수 있다. 버스(816)는 컴퓨팅 기기(800)의 하나 또는 복수 개의 컴포넌트에 통신적으로 커플링될 수 있다. 컴퓨팅 기기(800)는 또한 입력 기기 또는 출력 기기와 같은 복수 개의 외부 기기 또는 내부 기기를 포함할 수 있다. 예컨대, 도시된 컴퓨팅 기기(800)는 입력/출력(“I/O”) 인터페이스(818)를 구비하고, 하나 또는 복수 개의 입력 기기(820)로부터의 입력을 수신할 수 있거나 하나 또는 복수 개의 출력 기기(822)에 출력을 제공할 수 있다. 하나 또는 복수 개의 입력 기기(820) 및 하나 또는 복수 개의 출력 기기(822)는 I/O 인터페이스(818)에 통신적으로 커플링될 수 있다. 상기 통신적인 커플링은 임의의 적합한 방식을 통해 구현될 수 있다(예컨대, 인쇄 회로 기판을 통해 연결되는 것, 케이블을 통해 연결되는 것, 무선 전송을 통해 통신되는 것 등). 입력 기기(820)의 비 제한적인 예는 터치 스크린(예컨대, 터치 영역을 이미징하기 위한 하나 또는 복수 개의 카메라, 또는 터치로 인한 압력 변환을 검출하기 위한 압력 센서), 마우스, 키보드 또는 컴퓨팅 기기의 사용자의 물리적 동작에 응답하여 입력 이벤트를 생성하는데 사용될 수 있는 임의의 다른 기기를 포함한다. 출력 기기(822)의 비 제한적인 예는 LCD 스크린, 외부 모니터, 스피커 또는 컴퓨팅 기기에 의해 생성된 출력을 디스플레이하거나 다른 방식으로 제시하는데 사용될 수 있는 임의의 다른 기기를 포함한다.

[0258] 컴퓨팅 기기(800)는 프로그램 코드를 실행할 수 있고, 상기 프로그램 코드는 프로세서를 상기 도 1-7에 대한 하나 또는 복수 개의 단계를 실행하도록 구성한다. 프로그램 코드는 비디오 인코더(100) 또는 비디오 디코더(200)를 포함할 수 있다. 프로그램 코드는 메모리(814) 또는 임의의 적합한 컴퓨터 판독 가능한 매체에 상주하고, 프로세서(812) 또는 임의의 다른 적합한 프로세서에 의해 실행될 수 있다.

[0259] 컴퓨팅 기기(800)는 또한 적어도 하나의 네트워크 인터페이스 기기(824)를 포함할 수 있다. 네트워크 인터페이스 기기(824)는 하나 또는 복수 개의 데이터 네트워크(828)의 유선 또는 무선 데이터와 연결을 구축하기에 적합한 임의의 기기 또는 기기 그룹을 포함할 수 있다. 네트워크 인터페이스 기기(824)의 비 제한적인 예는 이더넷 네트워크 어댑터, 모뎀 및 유사한 기기 중 적어도 하나를 포함한다. 컴퓨팅 기기(800)는 네트워크 인터페이스 기기(824)를 통해 전자 또는 광학 신호의 형태로 메시지를 전송한다.

[0260] *일반적인 고려 사항*

[0261] 청구되는 주제에 대한 철저한 이해를 제공하기 위해, 본 문에서 많은 세부 사항을 설명하였다. 그러나, 본 분야의 통상의 기술자는, 이러한 세부 사항이 없는 경우에도, 청구되는 주제를 실시할 수 있음을 이해할 수 있다. 다른 예에서, 청구되는 주제가 모해해지는 것을 피하기 위해, 통상의 기술자가 이미 알고 있는 방법, 장치 또는 시스템을 상세히 설명하지 않았다.

[0262] 특별히 설명하지 않는 한, 본 명세서에서, “처리”, “계산”, “결정”, “식별” 또는 유사한 용어를 사용한 논의는 컴퓨팅 기기(예컨대 하나 또는 복수 개의 컴퓨터, 또는 유사한 전자 컴퓨팅 기기)의 동작 또는 과정을 가리키고, 이러한 기기가 컴퓨팅 플랫폼의 메모리, 레지스터 또는 다른 정보 저장 기기, 전송 기기 또는 디스플레이 기기에서 물리적 전자 또는 자기 양으로 나타내는 데이터를 조작하거나 전환한다.

[0263] 본 문에서 논의된 시스템은 임의의 특정된 하드웨어 아키텍처 또는 구성에 한정되지 않는다. 컴퓨팅 기기는 하나 또는 복수 개의 입력을 조건적인 결과로서 제공하는 임의의 적합한 컴포넌트 배치를 포함한다. 적합한 컴퓨

팅 기기는 저장된 소프트웨어를 액세스하는 마이크로 프로세서에 기반한 다용도 컴퓨터 시스템을 포함하고, 상기 소프트웨어는 컴퓨팅 시스템을 범용 컴퓨팅 장치로부터 본 주제의 하나 또는 복수 개의 실시예를 구현하는 전용 컴퓨팅 장치로 프로그래밍하거나 구성한다. 컴퓨팅 기기를 프로그래밍하거나 구성하기 위한 소프트웨어에서, 임의의 적합한 프로그래밍 언어, 스크립팅 언어 또는 다른 타입의 언어 또는 언어 조합은 모두 본 문에 포함된 교시를 구현하는데 사용될 수 있다.

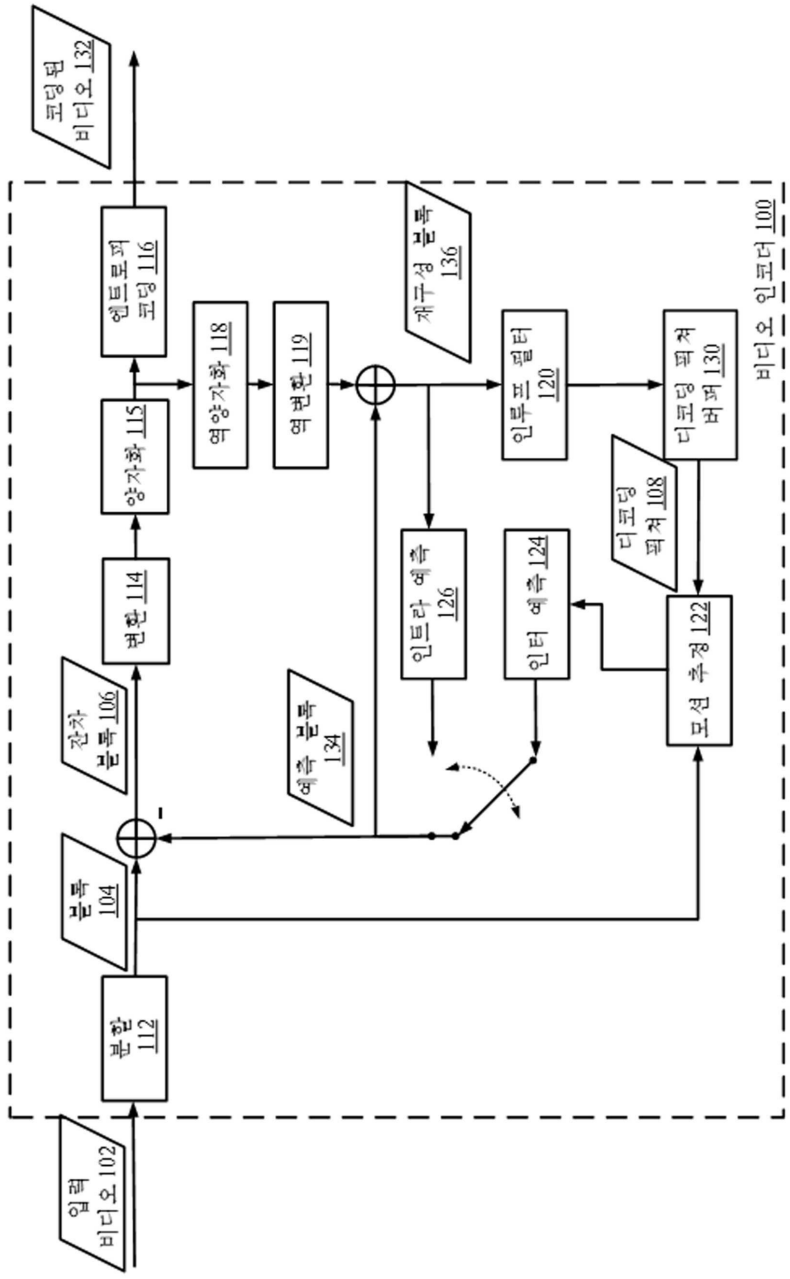
[0264] 본 문에서 개시된 방법 실시예는 이러한 타입의 컴퓨팅 기기의 동작에서 실행될 수 있다. 위의 예에서 제시한 단계의 순서는 개변될 수 있으며－－예컨대, 단계는 서브 블록으로 재배열, 조합 및/또는 분해될 수 있다. 특정된 단계 또는 과정은 병렬적으로 실행될 수 있다.

[0265] 본 문에서 사용되는 “적합” 또는 “구성”은, 부가 임무 또는 단계를 실행하는데 적합하거나 실행하도록 구성된 기기를 배제하지 않는, 개방적이고 포괄적인 언어이다. 또한, 단어 “기반”의 사용은, 하나 또는 복수 개의 열거된 조건 또는 값에 “기반”한 과정, 단계, 계산 또는 다른 동작은 실제로, 열거된 것 이외의 다른 조건 또는 값에 기반한 것일 수 있으므로, 개방적이고 포괄적인 것이다. 본 문에 포함된 제목, 리스트 및 번호는 다만 설명의 편의를 위한 것일 뿐, 한정을 의미하는 것이다.

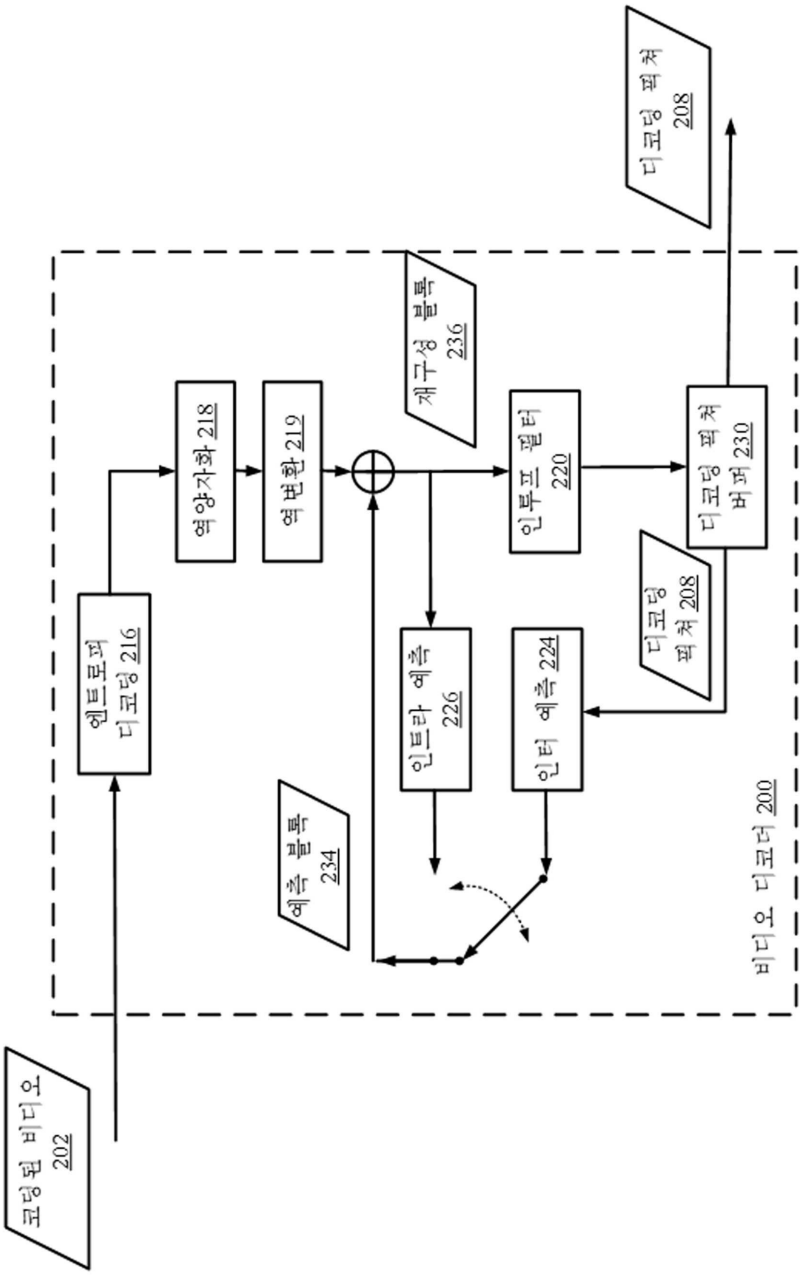
[0266] 비록 본 문의 주제에 대한 구체적인 실시예에서 본 문의 주제를 상세히 설명하였지만, 본 분야의 통상의 기술자는 전술한 내용을 이해한 후, 이러한 실시예에 대한 개변, 변화 및 동등한 것을 용이하게 수행할 수 있음을 이해해야 한다. 따라서, 이해해야 할 것은, 본 발명은 제한적이 아닌 예시의 목적으로 제출된 것이고, 본 분야의 통상의 기술자에게 있어서 자명한, 본 문의 주제에 대해 수행한 이러한 수정, 변화 및 추가 중 적어도 하나를 포함하는 것을 배제하지 않는다.

도면

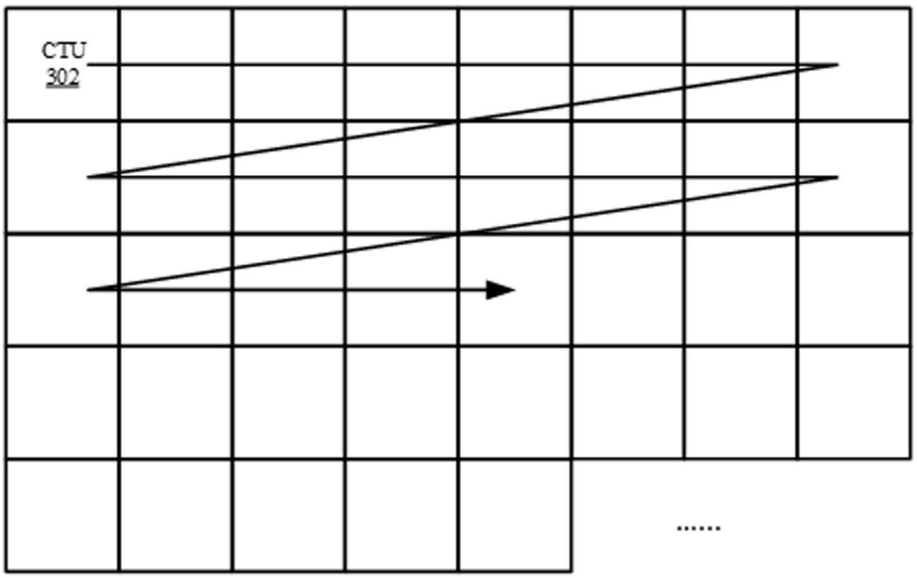
도면1


입력 비디오 102
분할 112
블록 104
잔차 블록 106
변환 114
양자화 115
엔트로피 코딩 116
코딩된 비디오 132
역양자화 118
역변환 119
예측 블록 134
인트라 예측 126
인터 예측 124
모션 추정 122
디코딩 픽처 108
인루프 필터 120
디코딩 픽처 버퍼 130
재구성 블록 136
비디오 인코더 100

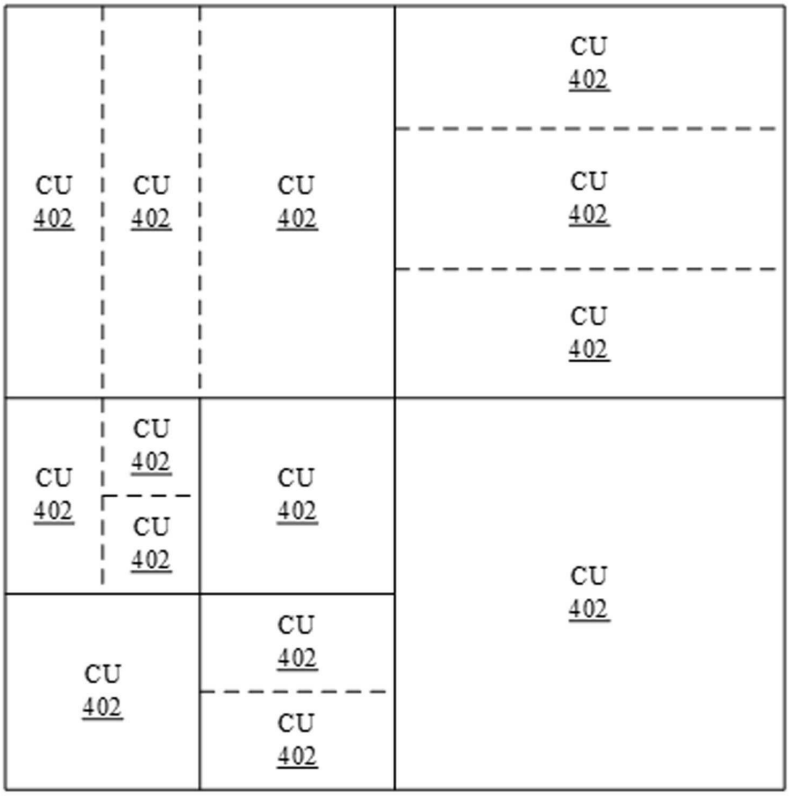
도면2


코딩된 비디오 202
엔트로피 디코딩 216
역양자화 218
역변환 219
재구성 블록 236
인트라 예측 226
인터 예측 224
예측 블록 234
인루프 필터 220
디코딩 픽처 버퍼 230
디코딩 픽처 208
디코딩 픽처 208
비디오 디코더 200

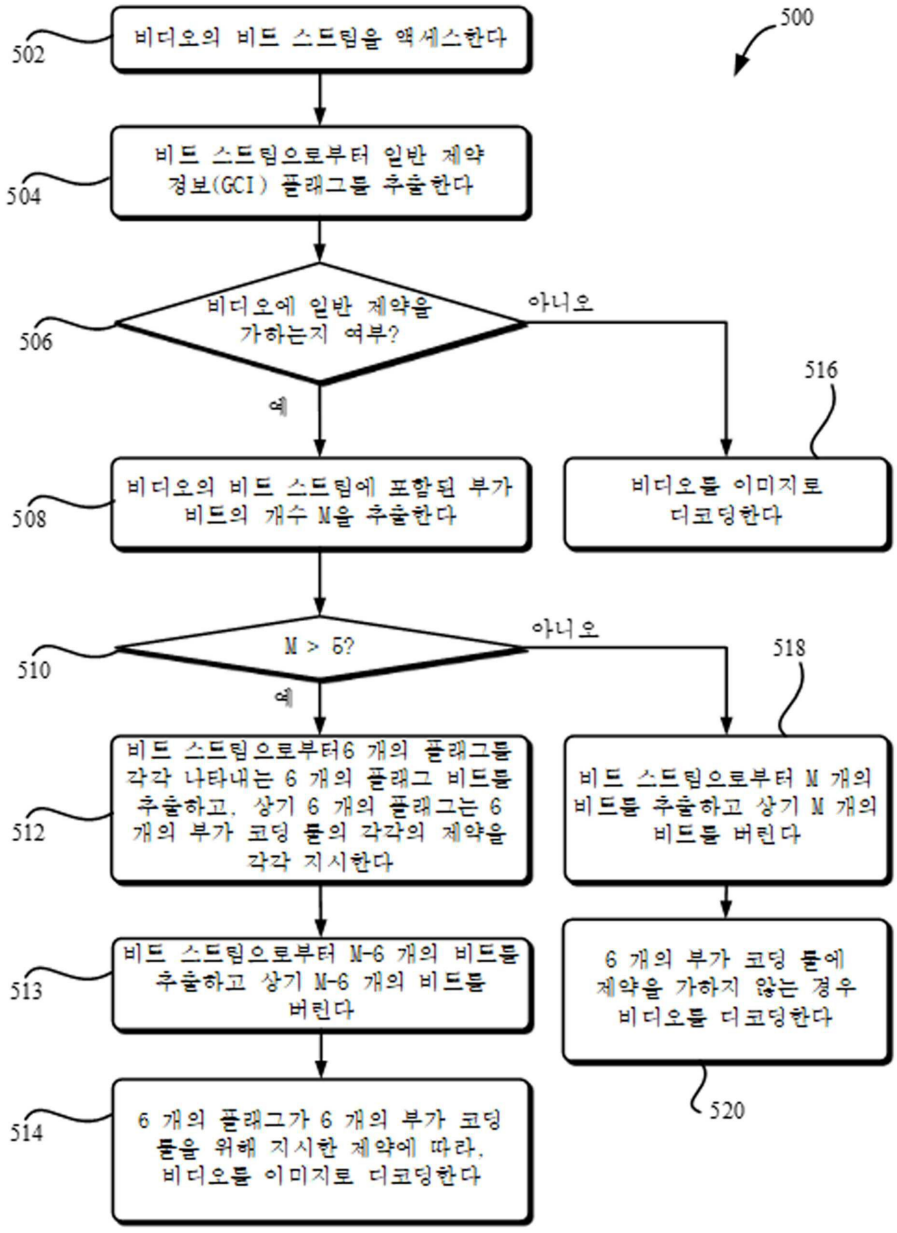
도면3



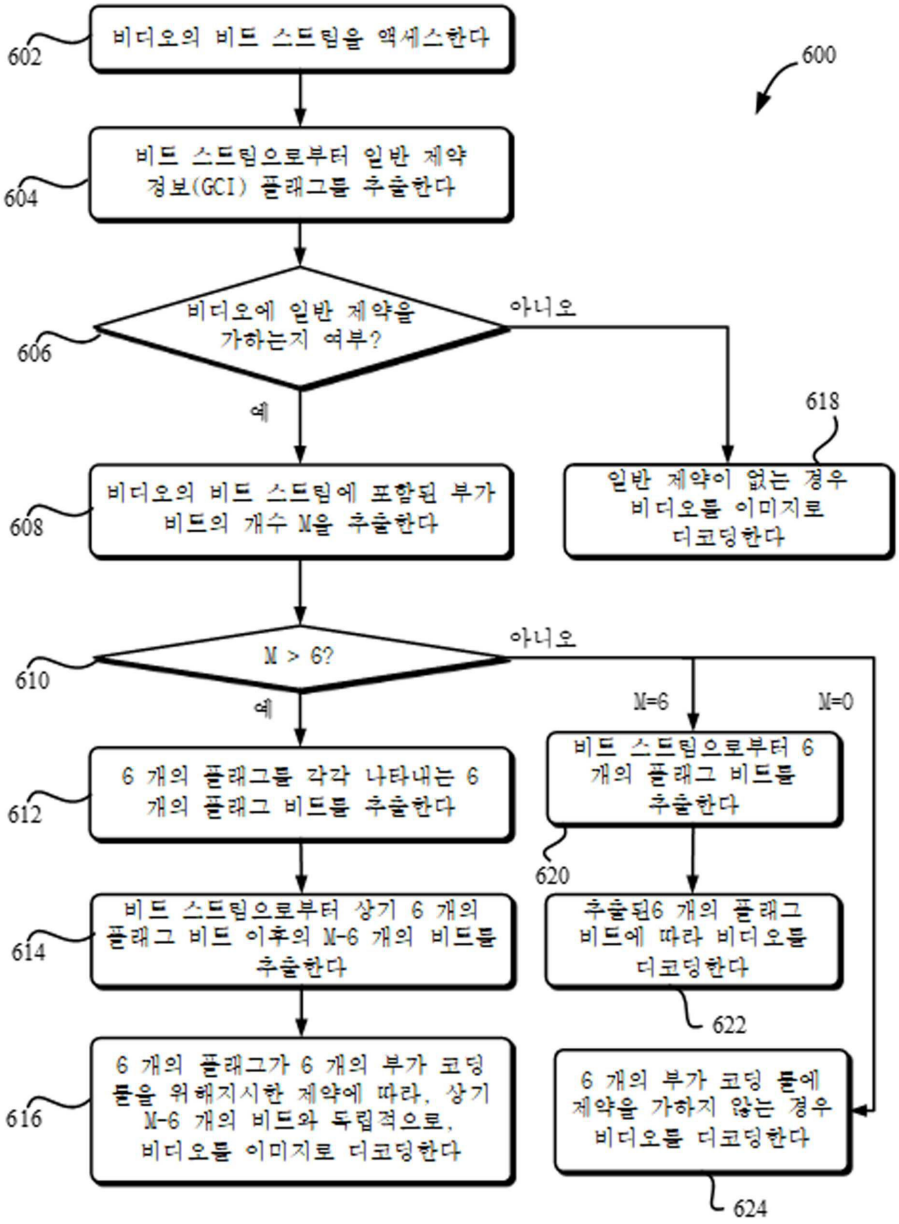
도면4



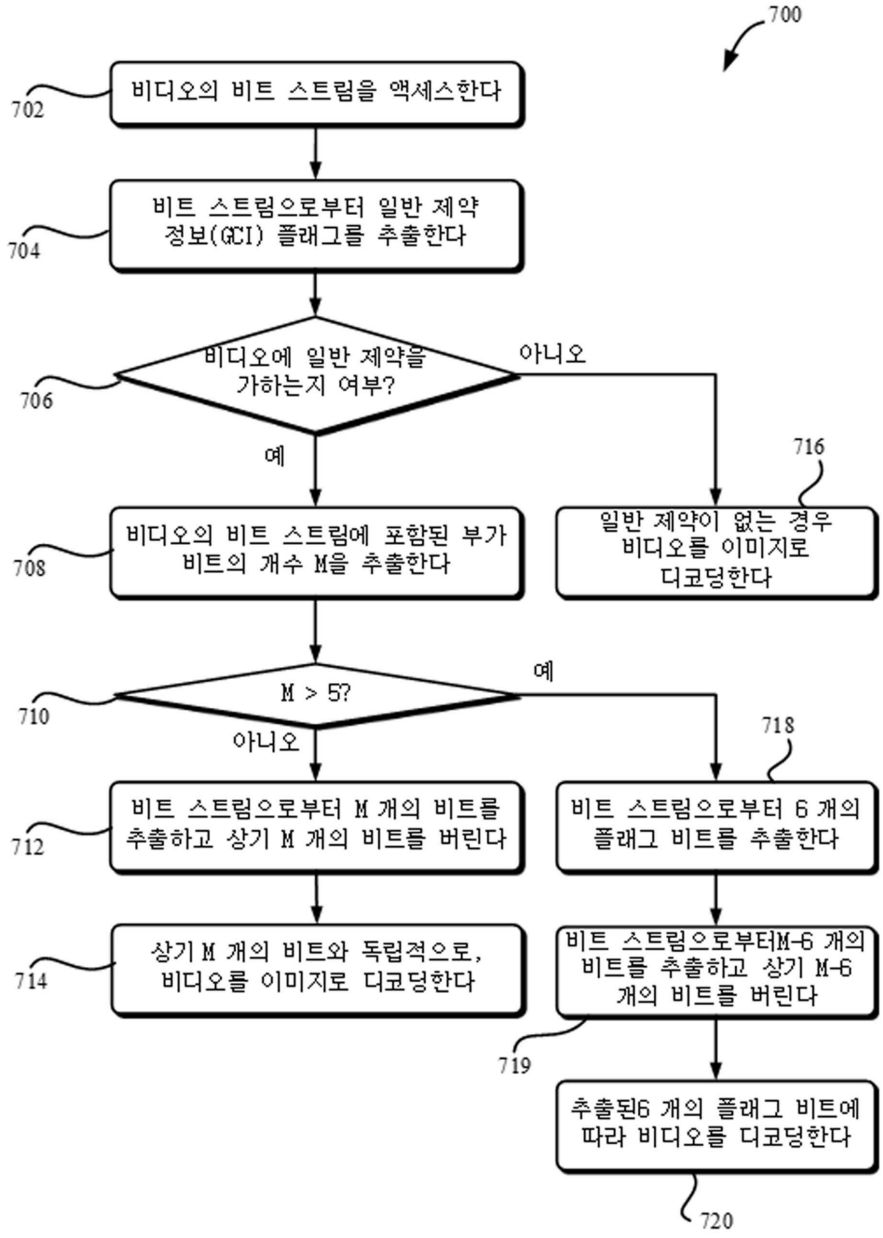
도면5


500
502
비디오의 비트 스트림을 액세스한다
504
비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출한다
506
비디오에 일반 제약을 가하는지 여부?
아니오
예
516
비디오를 이미지로 디코딩한다
508
비디오의 비트 스트림에 포함된 부가 비트의 개수 M을 추출한다
510
M > 6?
아니오
예
518
비트 스트림으로부터 M 개의 비트를 추출하고 상기 M 개의 비트를 버린다
512
비트 스트림으로부터6 개의 플래그를 각각 나타내는 6 개의 플래그 비트를 추출하고, 상기 6 개의 플래그는 6 개의 부가 코딩 툴의 각각의 제약을 각각 지시한다
513
비트 스트림으로부터 M-6 개의 비트를 추출하고 상기 M-6 개의 비트를 버린다
514
6 개의 플래그가 6 개의 부가 코딩 툴을 위해 지시한 제약에 따라, 비디오를 이미지로 디코딩한다
6 개의 부가 코딩 툴에 제약을 가하지 않는 경우 비디오를 디코딩한다
520

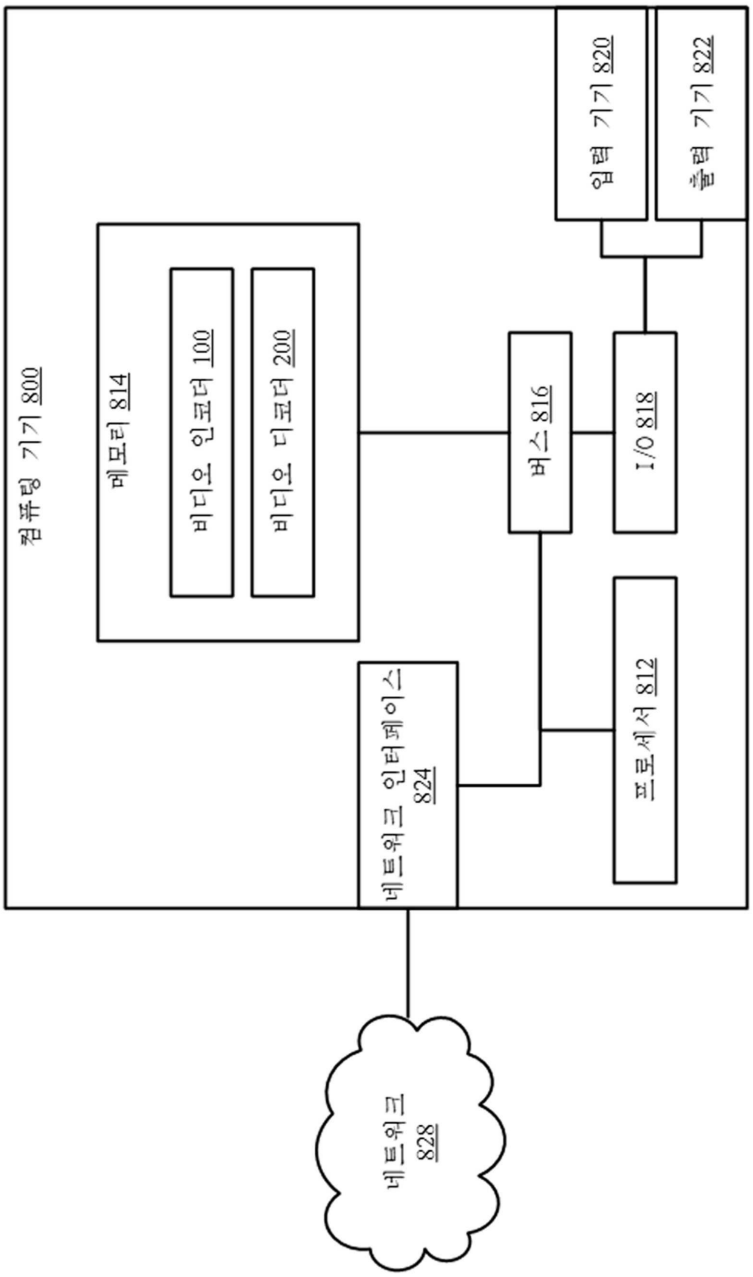
도면6


600
602
비디오의 비트 스트림을 액세스한다
604
비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출한다
606
비디오에 일반 제약을 가하는지 여부?
아니오
예
618
일반 제약이 없는 경우 비디오를 이미지로 디코딩한다
608
비디오의 비트 스트림에 포함된 부가 비트의 개수 M을 추출한다
610
M > 6?
아니오
예
M=6
M=0
612
6 개의 플래그를 각각 나타내는 6 개의 플래그 비트를 추출한다
비트 스트림으로부터 6 개의 플래그 비트를 추출한다
620
614
비트 스트림으로부터 상기 6 개의 플래그 비트 이후의 M-6 개의 비트를 추출한다
추출된6 개의 플래그 비트에 따라 비디오를 디코딩한다
622
616
6 개의 플래그가 6 개의 부가 코딩 툴을 위해지시한 제약에 따라, 상기 M-6 개의 비트와 독립적으로, 비디오를 이미지로 디코딩한다
6 개의 부가 코딩 툴에 제약을 가하지 않는 경우 비디오를 디코딩한다
624

도면7


700
비디오의 비트 스트림을 액세스한다
702
비트 스트림으로부터 일반 제약 정보(GCI) 플래그를 추출한다
704
비디오에 일반 제약을 가하는지 여부?
706
아니오
예
716
일반 제약이 없는 경우 비디오를 이미지로 디코딩한다
비디오의 비트 스트림에 포함된 부가 비트의 개수 M을 추출한다
708
M > 5?
710
예
아니오
718
비트 스트림으로부터 6 개의 플래그 비트를 추출한다
비트 스트림으로부터 M 개의 비트를 추출하고 상기 M 개의 비트를 버린다
712
상기 M 개의 비트와 독립적으로, 비디오를 이미지로 디코딩한다
714
비트 스트림으로부터M-6 개의 비트를 추출하고 상기 M-6 개의 비트를 버린다
719
추출된6 개의 플래그 비트에 따라 비디오를 디코딩한다
720

도면8


컴퓨팅 기기 800
메모리 814
비디오 인코더 100
비디오 디코더 200
버스 816
I/O 818
입력 기기 820
출력 기기 822
프로세서 812
네트워크 인터페이스 824
네트워크 828